TOPICS IN NETWORK OPTIMIZATION: THE STEINER TREE PROBLEM AND SEMICONDUCTOR MANUFACTURING

A Dissertation
Presented to
The Academic Faculty

By

Matias I. Siebert Sandoval

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
H. Milton Stewart School of Industrial & Systems Engineering

Georgia Institute of Technology

May 2020

TOPICS IN NETWORK OPTIMIZATION: THE STEINER TREE PROBLEM AND SEMICONDUCTOR MANUFACTURING

Approved by:

Professor Shabbir Ahmed
Co-Advisor
H. Milton Stewart School of Industrial & Systems Engineering
Georgia Institute of Technology

Professor George Nemhauser
Co-Advisor
H. Milton Stewart School of Industrial & Systems Engineering
Georgia Institute of Technology

Professor Mohit Singh
H. Milton Stewart School of Industrial & Systems Engineering
Georgia Institute of Technology

Professor Alejandro Toriello
H. Milton Stewart School of Industrial & Systems Engineering
Georgia Institute of Technology

Renan Garcia
Amazon

Date Approved: December 11, 2019

To my brother Tomás †,

to my grandpa Nano †,

to my advisor Shabbir †.

ACKNOWLEDGEMENTS

Getting my Ph.D. has been a long road, with its ups and downs. It was one of the best chapters of my life and I will always remember it with affection. It was the time that I, and my wife, decided to leave everything behind and live abroad. Being away from friends and family made us grow as persons, and made us stronger as a couple. This thesis, beyond the compilation of scientific contributions, represents the summary of all the work that my wife and I put into the last four and a half years.

First, I want to thank my beautiful wife Berni. You have been my partner from the start, when we began dating while in high school, 15 years ago. We were a couple of teenagers, and we have grown together. There are no words that can express how I feel about you. You are, by far, the best person I have had the pleasure to meet, and I will always try to love and protect you. I know you sacrificed a lot when we decided to move to the U.S., especially regarding your career as a bright organizational psychologist, but I am sure you will be able to resume your career sooner than later, because you are a very smart and hard working person. I know that when we are old and gray, we will look back at this time in our lives and will agree that, even though it was difficult sometimes, it was one of the best chapters of our lives. Finally, I wanted to thank you for giving me the best gift a husband can ask for, you made me a dad to our little Emma Sofia. I will be always thankful to have you both in my life, I cannot ask for anything else. This Ph.D. is not just for me, it is for all of us, at least that is how I feel about it. I love you and our beautiful daughter, and will always.

Secondly, I wanted to thank my family, my mom, my dad, my brother, and my grandma. You have always believed in me, and supported me in every decision I have made throughout my life. It has been very tough to be apart from you during all these years, I think about you every day. My parents raised me, they were always there for me and made sure I had a good life. You, as all of us may have your flaws, but I am very proud and thankful for the

parents that life gave me. I love you both very much. My little brother Joaquin, who stayed in Chile and takes care of my parents. I love you bro, I am very proud of the man you have become, and I want to let you know that you can always count on me, for anything. I will always be there for you whenever you need it, always. My grandma, Pichita, was always worried that me and Berni were OK, and made us delicious food when she came to visit us. I will always remember that you, with my grandpa Nano†, took care of me during the toughest time of our lives, when my little brother Tomás passed away. I will always have you in my heart. I would give everything to have my brother Tomás† and my Tata Nano† here to celebrate this achievement with us.

My advisors, Professors Shabbir Ahmed and George Nemhauser. They were the best advisors that I could have asked for. We spent several hours discussing problems, my ideas and proofs, and you guided me whenever I was lost. You were patient, and taught me several things to succeed as a researcher. I admire both of you, because you are the smartest people that I have ever met and worked with, but you were also humble. I think that your influence goes beyond my academic development; I look up to both of you, and I hope I can be more like you when I am older. During my last year as a Ph.D. student, Shabbir passed away. It was very hard for me, because Shabbir was very young, and this was very unexpected. I miss him very much, and I would have loved for him to be present in my thesis defense.

Friends I made during my stay at GT. First, I wanted to thank Alfredo Torrico, who was my first friend in Atlanta. I will always remember that you waited for me in the airport while studying for your comps. Also, we shared a lot of lunches, discussions, and parties. Second, I wanted to thank Ramon Auad and Sebastian Perez. We become friends during my third year at GT, and we also shared several lunches, discussions and parties. Felipe Lagos, who was my office mate, and we also did an internship together at Amazon. There are several other friends that I made during my Ph.D., including Adolfo Rocco Adrian Rivera, Seyma, Yulia, Beste, Asteroide Santana, Mariana Almeida, John Connelly, Maira, Yeyo,

Hine, Camila Apablaza, Camila Albornoz, Tony Yaacoub, German, Idil, Damian Reyes, Ethan Mark, Francisco Castillo, Fran Villegas, Daniela Hurtado, Alejandro Carderera, Ian, Guido Lagos, Alvaro Lorca, Mathias Klapp, among many others. I am sorry if I missed someone, but as you can see, the list is quite long.

My friends from my home country, Chile. In particular, I want to thank my group of friends "Pollos Hermanos", which is composed of friends I met during high-school. You have always been supportive, and encouraged me to pursue my dreams. Also, we always hang out when I go to visit Chile, and I hope we can continue our friendship for many years to come. Secondly, I want to thank my friends from college. We spent hours studying and working on homework assignments during my bachelors and masters. I learned a lot from you, and I think that in some way or another, you had a huge impact in my decision to get a Ph.D.

The ISyE department for giving me the tools and support to pursue my Ph.D. Looking back, I think I made the best decision to get my Ph.D. at Georgia Tech. Also, I want to thank the professors I met and interacted with at Georgia Tech. The quality of the faculty is incredible, and I think I was fortunate to take classes and work with such amazing researchers. Finally, I want to thank Alan Erera and Amanda Ford for all the support given when I needed help with all the bureaucratic aspects of Ph.D. life.

I want to thank the members of my thesis committee, professors Alejandro Toriello and Mohit Singh, and Dr. Renan Garcia. I really appreciate the time you took to be a part of my committee, and also all your valuable comments and questions during my thesis proposal and my thesis defense.

Finally, I wanted to thank Conicyt (Chilean National Commission for Scientific and Technological Research) through the Doctoral Fellowship program "Becas Chile", (grant number 72160393), the scholarship sponsor from Chile. None of this would have been possible without the investment of the Chilean government. I hope I will be able to reciprocate accordingly."

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis covers two very different topics related to problems defined on graphs. The first is a fundamental graph optimization problem called the Steiner tree problem. The second is an applied problem in semiconductor manufacturing.

In the first part of this thesis, we propose a new approach to solve the Steiner tree problem when the number of terminal nodes is fixed. The Steiner tree problem is a classical network design problem. We are given a graph $G = (V, E)$, a set of terminal nodes $R \subseteq V$, and a non-negative cost vector for the edges in E. The problem is to find the minimum weight tree in G that spans all nodes in R. We present a set of integer programs (IPs) for the Steiner tree problem with the property that the best solution obtained by solving all, provides an optimal Steiner tree. Each IP is polynomial in the size of the underlying graph and our main result is that the linear programming (LP) relaxation of each IP is integral so that it can be solved as a linear program. However, the number of IPs grows exponentially with the number of terminals in the Steiner tree. As a consequence, we are able to solve the Steiner tree problem by solving a polynomial number of LPs, when the number of terminals is fixed.

To address the latter issue, we propose a local-search based algorithm to solve the problem for big instances. First, we propose a dynamic programming algorithm to solve each IP efficiently. Then, we provide a characterization of the neighborhood of each IP. Finally, we propose a simulated annealing framework to solve the problem. We present computational results for a large set of instances in the SteinLib library, comparing our proposed approach with state-of-the-art algorithms to solve the directed version of the Steiner tree problem. We study over 800 instances from the SteinLib library, and we show that the solution quality of the proposed approach surpasses the quality of the solution of the state-of-the-art algorithms.

In the second part of this thesis, we study the semiconductor manufacturing problem,

which is a highly complex and dynamic re-entrant process where wafers go through several processing steps, entering the same group of machines multiple times. We propose a fluid model lot dispatching policy that iteratively optimizes lot selection based on current work-in-progress (WIP) distribution of the entire system. A fluid model is an approximation technique used to model and study the dynamics of a stochastic queuing network framework. Furthermore, we propose to split the decision policies into two phases in order to include travel time information into the dispatching and targeting decisions. We provide simulation results for a prototype facility that show that our proposed policies outperform commonly used dispatching rules in throughput, machine utilization, and machine target accuracy.

CHAPTER 1
INTRODUCTION AND BACKGROUND

Network optimization is a very important and wide class of optimization problems. These problems arise from different real-world applications such as transportation and logistic systems [65, 101, 91, 105], power-systems [7, 38, 71, 70], communication networks [2, 80, 98], and social networks [56, 14, 19], among others.

There are network problems which are easy to solve, i.e., solvable in polynomial time, such as the shortest path problem [28], minimum spanning tree [61, 85], maximum flow [42], minimum cut [27], maximum weighted matching in bipartite graphs [64], and maximum weighted matching in general graphs [35], and so forth. On the other hand, there are several network optimization problems which are difficult to solve, meaning that no polynomial-time algorithm is known to solve and none is likely to exists, i.e., they are NP-Hard. Among these problems we can mention the traveling salesman problem [55], the vehicle routing problem [66], and the fixed charge multicommodity network problem [74], among others. This thesis studies two of the difficult problems in networks, the Steiner tree problem, and the semiconductor manufacturing problem.

1.1 Preliminary definitions

Network and graph definitions of terminology used within this thesis are provided in Section A.1 of Appendix A.

1.1.1 Definitions

A set is a collection of elements. Given a set S, we define $p = \{S_1, S_2, \ldots, S_{|p|}\}$ to be a partition of set S if, $i)$ $S = \bigcup_{i=1}^{|p|} S_i$; and $ii)$ $S_i \cap S_j = \emptyset$ for all $i, j \in \{1, \ldots, |p|\}, i \neq j$. We assume that $|S_i| \geq 1$ for all $i \in \{1, \ldots, |p|\}$. We say that p is a proper partition of S if

$|p| \geq 2$.

A family $\mathcal{C}$ of sets is called laminar if for every $A, B \in \mathcal{C}$ we have $A \subseteq B$ or $B \subseteq A$ or $A \cap B = \emptyset$. Every laminar family C has a unique forest representation, where each tree of such a forest is a rooted tree [37]. This implication goes both directions, i.e., every forest composed of a set of rooted trees has a unique associated laminar family.

For an undirected graph G, the contraction of edge $e = \{i, j\} \in E$ is the operation of removing edge e from G, and merging edges i and j into edge l. For all $e' \in \delta(i) \backslash \{e\}$, we replace the endpoint i with l, and for all $e' \in \delta(j) \backslash \{e\}$, we replace the endpoint j with l.

Finally, consider the following general form of linear program (LP), or linear optimization problem, $P := \min_{x \in X} c^T x$ where $X = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$, with $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$. P is said to be a linear program, since the variables are continuous, and the objective function and constraints are linear. Suppose $X \neq \emptyset$, then $\hat{x} \in X$ is said to be a solution to P, and the set of non-zero components of $\hat{x}$ is said to be the support of the solution. When a subset of the variables are forced to be integer, the problem is said to be a mixed integer program (MIP). The linear relaxation, or LP relaxation, of a MIP is when we consider the integer variables of the MIP as continuous variables.

1.2 Steiner tree problem

In this thesis we consider two variants of the Steiner tree problem, the undirected Steiner tree problem, and the directed Steiner tree problem. Throughout the rest of this thesis, we will refer to the undirected variant of the problem as the Steiner tree problem, while we will refer to the directed variant of the problem as the directed Steiner tree problem. All the network and graph terminology used throughout this section is defined in Section A.1 of Appendix A.

The Steiner tree problem in graph G is defined by the tuple (V, E, R, c), where V is the set of nodes of graph G, E is the set of edges of graph G, $R \subseteq V$ is the set of terminal nodes, and c is the cost vector of the edges of graph G, where $c_e \geq 0$ for all $e \in E$. The

problem is to find a minimum weight tree $T \subseteq G$ that spans all the nodes in R, where the weight of T is given by the sum of the edge weights over all edges in T. We assume that for all $t, t' \in R$, there is a path connecting t and t' in G, otherwise the problem is infeasible.

The directed Steiner tree problem in graph D is defined by the tuple (V, A, R, r, c), where V is the set of nodes of D, A is the set of arcs of D, R is the set of terminal nodes, $r \in V$ is the root node, and c is the cost vector of the arcs of D, where $c_a \geq 0$ for all $a \in A$. The problem is to find a minimum weight arborescence rooted in r that spans all nodes in R, where the weight of an r-arborescence is given by the sum of the weight of all arcs in such directed tree. We assume that there is at least one path from r to every node in R, in graph D, otherwise the problem is infeasible.

For both variants of the problem, we assume, without loss of generality, that the cost vector is strictly positive since we can always contract the zero cost edges, or arcs, and keep at least one optimal solution of the problem.

The Steiner tree problem is NP-Hard [55], in fact, it is even NP-Hard to find approximate solutions whose cost is within a factor $\frac{96}{95}$ of the cost of an optimal solution [6, 20].

An important line of research has been to study integer linear programming (IP) formulations for this problem. There are several IP formulations of this problem, and they are often studied based on their LP relaxation via the integrality gap of the formulation. The integrality gap of a formulation is the ratio between an optimal integer solution and the value of the LP relaxation; the lower the integrality gap, the better the formulation is considered. In [47], the authors provide a catalog of Steiner tree formulations, and show the equivalence of some of these formulations. In [46] the author studies the vertex-weighted version of the undirected Steiner tree problem, and presents a complete description of the polytope when the graph is *series-parallel*. By projecting this formulation, some facet-defining inequalities for the Steiner tree polytope are obtained. In [81], the authors compare different formulations of the Steiner tree problem in terms of the strength of their linear relaxations (LP), and propose a new polynomial size formulation with a stronger LP relaxation than

the ones analyzed in their paper.

In every Steiner tree, we can pick an arbitrary node $r \in R$ and find a direction of the edges, such that we can construct a unique arborescence rooted at r whose leaf nodes correspond to nodes in $R\backslash\{r\}$, which we also refer to as R^r. Consequently, the Steiner tree problem can be modeled as a directed variant of the problem. A well studied formulation that uses this fact is the *Bidirected Cut* formulation [36, 109].

$$\min \sum_{a \in A} c_a z_a$$

$$\sum_{a \in \delta^-(S)} z_a \geq 1 \qquad \forall S \subseteq V\backslash\{r\} : S \cap R^r \neq \emptyset \tag{1.1a}$$

$$z_a \in \{0, 1\} \qquad \forall a \in A \tag{1.1b}$$

Where z_a is 1 if arc a is included in the solution, 0 otherwise. This is a compact and simple formulation, whose integrality gap is known to be at most 2, but is believed to be close to one. The best known lower bound on the integrality gap for this formulation is $\frac{36}{31} \approx 1.16$ [13]. An upper bound on the integrality gap for this formulation has been found for special cases. For *quasi-bipartite* graphs, the integrality gap is upper bounded by $\frac{3}{2}$ [86], and for *claw-free* graphs it is upper bounded by $\ln(4)$ [39].

Another well studied formulation is the so called *Hypergraphic* formulation [60, 84, 106].

$$\min \sum_{C \in \mathcal{K}} \text{cost}_C x_C$$

$$\sum_{C \in \mathcal{K}} x_C (|R(C) \cap S| - 1)^+ \leq |S| - 1 \qquad \forall S \subseteq R, S \neq \emptyset \tag{1.2a}$$

$$\sum_{C \in \mathcal{K}} x_C (|R(C)| - 1)^+ = |R| - 1 \tag{1.2b}$$

$$x_C \in \{0, 1\} \qquad \forall C \in \mathcal{K} \tag{1.2c}$$

This formulation has one variable, x_C for each *component* C of the graph, where a *component* is a tree whose leaves correspond to terminal nodes. $\mathcal{K}$ is the set of all components, and cost_C is the cost of component $C \in \mathcal{K}$. The number of variables and constraints for this formulation are proportional to the number of *components*, which is exponential in the number of terminal nodes. On the positive side, we have that the *Hypergraphic* formulation is stronger than the *Bidirected Cut* formulation [84]. The integrality gap of the *Hypergraphic* formulation is lower bounded by $\frac{8}{7} \approx 1.14$ [60] and upper bounded by $\ln(4)$ [48]. Moreover, in [39], the authors show that in *claw-free* graphs, both formulations are equivalent. On the down side, solving the *Hypergraphic* formulation is strongly NP-Hard [48]. To address this problem, researchers have proposed to solve it by only considering components with at most k leaves, where k is a fixed value. This considerably reduces the number of constraints and variables, and the solution of this restricted formulation is proven to be within a factor of ρ_k of an optimum solution, where $\rho_k \leq 1 + \frac{1}{\lfloor \log_2(k) \rfloor}$ [10]. A complete review of the *Hypergraphic* formulation and its variants can be found in [15].

In [30], the authors propose a Dynamic Programming (DP) algorithm that finds an optimal solution for the Steiner tree problem in $\mathcal{O}(n^3 + n^2 2^b + n3^b)$ time, where $n = |V|$, $m = |A|$, and $b + 1 = |R|$. The result of this paper suggests that, for a fixed number of terminals, there exists a polynomial size LP formulation of the Steiner tree problem [75], which is the motivation for the results presented in this thesis. The main drawback of this approach, common in DP algorithms, is that it only finds a solution when the algorithm terminates. For real-world size instances, this algorithm is not practical since its running time is exponential in the number of terminals.

For the directed Steiner tree problem, there are not any known constant factor approximation algorithms. Actually, in [50], the authors show that the directed Steiner tree problem admits no $\mathcal{O}\left(\log^{2-\epsilon}(|R|)\right)$ algorithm, unless $NP \subseteq ZTIME(n^{polylog(n)})$. The best known approximation ratio for the directed Steiner tree problem is $\mathcal{O}\left(|R|^{\frac{1}{t}}\right)$ for $t > 1$, where t is the level of the constructed tree. This approximation ratio corresponds to the algorithm

presented in [16]. This algorithm constructs low density directed Steiner trees in polynomial time, where the density of a tree is defined as the ratio of the cost of the tree over the number of terminals of the tree.

For the Steiner tree problem, there are several time efficient algorithms to reduce the instances size, keeping at least one optimal solution. The milestone of this line of research is the Ph.D. thesis of C.W. Duin [34]. The reduction algorithms, also referred as reduction tests, can be categorized into two classes as stated in [82], the alternate-based tests and the bound-based tests. The first tests use the existence of alternative solutions, for instance, a test may discard an edge, or vertex, of the graph since there is at least one solution with a lower cost that does not contain such an edge, or vertex. On the other hand, the bound-based tests compare a lower bound of an optimal solution under the assumption that a part of the graph is included (excluded). If such lower bound is greater than a known upper bound of an optimal solution to the problem, then we conclude that such part of the graph has to be excluded (included). More details about these reduction techniques for general graphs can be found in [33, 32, 31, 82, 83]. There are also reduction techniques designed for particular types of graphs, that take into account the structure of such graphs. For instance, for rectilinear graphs the authors of [108, 103] present reduction techniques that take into account the particular topology of rectilinear graphs. All of the previously mentioned algorithms that are used to reduce the size of an instance are usually very powerful, and in several cases they can even find optimal solutions [59].

There are several papers that present different solution techniques to solve the Steiner tree problem to optimality with low execution times. In [58] the authors present a branch-and-cut procedure which is based on an IP formulation for the directed version of the problem. In such branch-and-cut framework the authors apply reduction tests, separations algorithms and primal heuristics. In [83], the authors proposed a branch-and-bound algorithm that combines alternate and bound based reduction tests, together with lower and upper bound computations. The authors invest more time in each node of the branch-and-bound

tree, with the goal of computing good lower and upper bounds, and keep the search tree as small as possible. In [3], the authors propose a branch-and-ascent framework applied to the Bidirected cut formulation. At the root node of the search tree, the algorithm computes a dual solution to the LP relaxation of the Bidirected cut formulation using Wong's algorithm [109]. Then, the algorithm branches on a Steiner node whose value is fractional by, either making such node a terminal node, or by removing the node from the graph. Each of the new nodes of the search tree corresponds to a new Steiner tree problem, one with an extra terminal, the other with a fewer Steiner node. Each of such nodes is solved again using Wong's algorithm. In [52], the authors propose a DP algorithm based on Dreyfus and Wagner's DP algorithm [30], but using a generalization of the speed up concept used in Dijkstra's algorithm to compute a shortest (s, t)-path, where only a few nodes are labeled before the node t is labeled permanently [51]. This concepts is used to compute just a few of the Steiner sub-trees of Dreyfus and Wagner's algorithm, leading to huge speed ups. Since all of the solution algorithms and reduction tests use the fact that the graph is undirected, then, in most cases, they cannot be extended to the directed case [88].

In contrast, there are not many papers in the literature related to computational experiments for the directed Steiner tree problem. In [107], the authors present two approximation algorithms, with $\mathcal{O}(|R|)$ and $\mathcal{O}(\sqrt{|R|})$ approximation ratios respectively. The authors present computational experiments, creating directed instances from over 900 undirected instances of the SteinLib library. All the directed graphs created by the authors keep at least one optimal solution of the original instance; consequently, they know in advance the value of an optimal solution. They compare their proposed algorithms against four benchmark algorithms used in practice [16, 100, 109]. The authors conclude that their proposed algorithms outperform the rest of the studied algorithms in terms of solution quality, while having similar execution times.

1.2.1 Our contributions

Our main contributions for the Steiner tree problem are the following.

1. We present a set of integer programs (IPs) for the Steiner tree problem with the property that the best solution obtained by solving all IPs provides an optimal Steiner tree. Each IP corresponds to the problem of finding an optimal best Steiner tree with a specific tree structure, where the structure is defined by the way the paths share arcs from the root node to each one of the terminal nodes.

2. Each IP is polynomial in the size of the underlying graph and our main result is that the linear programming (LP) relaxation of each IP is integral so that it can be solved as a linear program.

3. We propose a dynamic programming algorithm to solve each IP efficiently. Computational results show that the proposed algorithm is much faster than solving the LP relaxation of each IP using commercial solvers as Gurobi or CPLEX.

4. However, the number of IPs grows exponentially with the number of terminals in the Steiner tree. To address the latter issue, we propose a local-search based algorithm to solve the problem for big instances. We compare our proposed local-search approach with state-of-the-art algorithms to solve the directed version of the Steiner tree problem. The proposed approach outperforms such algorithms in solution quality, while solving the problem in a few seconds for most of the studied instances, and a few minutes for larger instances.

5. Finally, the proposed approach can be used to improve the solutions obtained by solving the problem with any algorithm that does not deliver an optimal solution; for instance, the solution delivered by any heuristic or approximation algorithm. By identifying the tree structure of the solution, it can solve the sub-problem for such

structure, and either find a better solution, or prove that the solution is optimal for the corresponding tree structure.

1.3 Semiconductor manufacturing

Wafer fabrication in semiconductor manufacturing is a very complex and dynamic re-entrant process. Wafers go through several of processing steps, entering the same processing group of machines more than once. An automated material handling system (AMHS) moves wafers from step to step, using overhead hoist transport (OHT) vehicles. Since wafers are high-value items, and machines are very expensive, manufacturers keep machine utilization as high as possible. Consequently, machine scheduling is a significant concern for managers given its impact in manufacturing productivity.

When demand for the AMHS is significantly higher than expected, the AMHS itself may become a production bottleneck due to vehicle congestion and blocking. Figure 1.1 illustrates a real-world problem. The manufacturer's AMHS was initially designed to handle a higher workload than the production process could produce, however, as production capacity increased, the AMHS became a bottleneck causing actual throughput to fall below expected throughput. This resulted in a loss of US$100 million between Month 1 and Month 2.

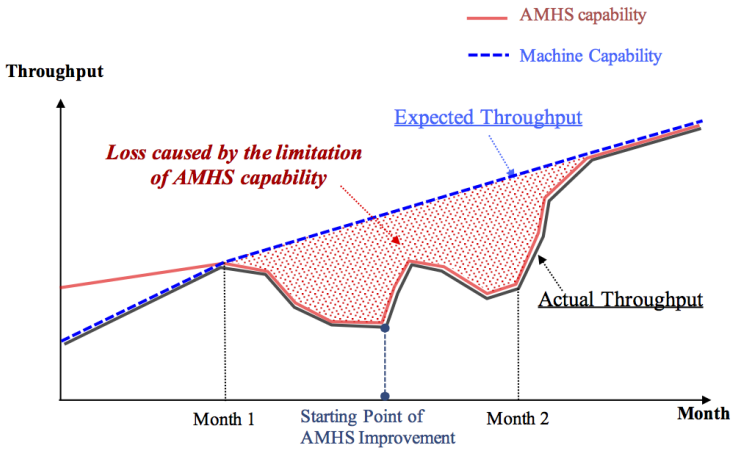


Figure 1.1: A loss of throughput by the shortage of AMHS capability

Given the complexity of wafer production, most wafer fab scheduling is based on simple greedy policies that decide which wafer lot will be processed next when a machine becomes available. In practice, other decisions need to be made. We summarize the main decisions as follows:

- *Lot Targeting*: Once a machine finishes processing a lot (batch) of wafers, the system must determine which machine will next process the lot. This decision considers the current workload of all machines with the correct process step, travel times to those machines from the current location, and other system-wide factors.

- *Lot Dispatching*: Once a machine becomes available, the system must decide which lot to pull from a storage location for processing at this machine. This decision considers, for each possible lot, travel time from storage to the machine, delay penalties, and other system-wide factors.

1.3.1 Classic lot scheduling

Lot scheduling is a classic research topic in semiconductor manufacturing. In [111], the authors provide a comprehensive survey of job shop scheduling problems applied to material handling systems, and in [79], the authors give a deep description of the semiconductor manufacturing process, the typical scheduling problems faced in these systems, and some of the important solution techniques to address these problems. Pre-allocation of work-in-progress (WIP) has been considered a solution because it can attain high machine utilization. In [45], the authors test the performance of pre-allocation in make-to-stock manufacturing systems with variable batch size and sequence-dependent setup. Due to the difficulty of estimating the proper WIP level for pre-allocation, most fabs employ instant decision making rules based on system conditions, called dispatching rules. In [72], the authors, who develop several dispatching rules, prioritize Front Opening Unified Pods (FOUPs) using different system characteristics that are expected to improve different per-

formance metrics. Later, in [67], a dispatching rule that aims to balance machine utilization is proposed.

Dispatching rules do not always perform as expected. In [78], the authors build industry-scale simulation models to compare the performance of various dispatching rules including the ones developed in [72], and in [67]. Selecting a job with the shortest processing time (SPT) is optimal for a single machine with deterministic processing time. In [90], experimental results where SPT is not optimal in realistic settings are presented. [89] also finds that dispatching rules based on the critical ratio of a job (the ratio of remaining time until due date to total remaining processing time) are affected by the assumptions regarding waiting and transfer time in manufacturing.

The literature has evolved to create new dispatching policies which are based on a combination of simpler dispatching rules. For instance, [24] and [25] combine multiple dispatching rules by calculating a linear combination of scores from individual dispatching rules. In [102] and [18], the authors apply different dispatching policies depending on system conditions. In [63], rework strategies and dispatching rules are combined, and in [110], coupling of policies for dedicated and non-dedicated machines are studied. In [68], the authors propose an integrated release and dispatch policy; they allocate production time and queuing time for each process step for every product, the first is used to define the time and product to release to the system, and the second to decide the dispatching policy. In [76], the authors perform simulations to collect data about the relationship between the simple dispatching rules and the system's performance in mean flow time, slack time, and total remaining processing time, and then they use a competitive neural network to learn, and decide which is the appropriate policy to use according to the status of the system. More recently, in [69], a heuristic is presented to determine both, the next wafer to be processed, and the starting time of the selected wafer, with the objective of meeting the due date of the corresponding lot. In their setting, the authors equally penalize for being early or being late. Their approach, proposes a decision tree that bases its decision on the due

date and the processing time of each job in the queue.

1.3.2 Scheduling based on queuing networks and fluid models

Queuing networks are commonly used to study the dynamics of complex systems in steady-state. Lot scheduling can be viewed as a queuing network problem, since lots move among service centers, each one with its own queue. Fluid models are derived from an underlying queuing network system, where the stochastic queuing system is replaced by a deterministic continuous model [44]. While traditional queuing models allow the evaluation of system performance under a particular set of policies, fluid models allow the dynamic optimization of these policies. In this section, we briefly review traditional queuing network models and follow with a discussion on alternative fluid models.

According to the survey paper of queuing theory [94], the first reported work applying a queuing network model to a dynamic job shop was proposed in [53]. The author studies a job shop network with a Poisson distributed arrival process and exponentially distributed processing times and derives a stationary equilibrium. The assumptions are simple and leave out important aspects of a real situation. Some extensions have been made to the original model proposed in [53]. In [62], the author proposes a decomposition approach to handle queuing networks with generally distributed inter-arrival times and processing times. This approach is applied to a manufacturing system in [93].

Some research has related queuing networks and fluid models. For example, [26] proves that a scheduling policy in a queuing network is stable if the underlying fluid network is stable. We note that while a fluid model is a pure deterministic model, queuing networks handle the stochasticity of a system. [17] was the first work to propose a dynamic scheduling based on a fluid model approach. In [22], the authors are the first to apply the algorithm based on fluid models proposed by [17] to the semiconductor manufacturing problem.

1.3.3 Our contributions

Our main contributions in this topic are summarized below.

1. We propose a new fluid-model based lot dispatching policy that iteratively optimizes lot selection based on current WIP distribution of the entire system.

2. Furthermore, we propose to split the decision policies into two phases in order to include travel time information into the dispatching and targeting decisions.

3. We provide simulation results for a prototype facility that show that our proposed policies outperform commonly used dispatching rules in throughput, machine utilization and machine target accuracy.

CHAPTER 2
A LINEAR PROGRAMMING BASED APPROACH TO THE STEINER TREE PROBLEM WITH A FIXED NUMBER OF TERMINALS

The work presented in this chapter has already been published [95].

2.1 Introduction

In this chapter, we propose a new approach for the Steiner tree problem, when the number of terminals is fixed. We select an arbitrary terminal node as a root node, and we study the way the paths from this root node, to the rest of terminal nodes, share arcs. We observe that, in every Steiner tree, the underlying structure of the way the paths share arcs is laminar. Based on this observation, we propose a new set of formulations for the problem, which is polynomial in size when the number of terminal nodes is fixed.

2.1.1 Our contributions

Our contributions are the following.

- We propose a set of independent IPs with the property that the best solution obtained by solving all of them, provides an optimal Steiner tree.
- Each IP is polynomial in the size of the underlying graph and our main result is that the LP relaxation of each IP is integral. Furthermore, the set of LPs can be solved in parallel.
- The drawback is that the number of LPs grows exponentially with the number of terminal nodes so the complete algorithm is polynomial only for a fixed number of terminals.

- In contrast to the DP approach proposed in [30], each IP delivers a feasible solution to the problem, and therefore we get a new solution every time we solve a new IP, which implies that we may get good solutions by solving just a subset of the IPs.

The remainder of this chapter is organized as follows. Section 2.2 analyzes the structure of Steiner trees, and also introduces definitions used in the chapter. Section 2.3 presents the proposed model, the proof of integrality, and an analysis of the size of the problem. In Section 2.4 we present computational experiments obtained by solving the Steiner tree problem using our proposed formulation. Section 2.5 gives conclusions.

2.2 Solution Structure

As mentioned in section 1.2, several formulations use the fact that every Steiner tree contains an r-arborescence, some of which use a flow-based formulation, and others a cut-based formulation. Both of those common approaches work with the directed graph version D of G. In our approach, we exploit two main properties that every solution must have.

1. We can always pick an arbitrary node $r \in R$ as our root node, and find the min-cost r-arborescence that spans R^r, also referred as R^r. Moreover, this r-arborescence can be modeled as a Min-cost Multi-Commodity Network Flow, where we have one commodity for every node in $R \backslash \{r\}$, and we want to send 1 unit of flow from r to the corresponding node $i \in R^r$. In this setting, we do not consider variable costs, and we consider the arcs to have infinite capacity.

2. Since the solution must correspond to an r-arborescence, there must be exactly one path from the root node r to each one of the nodes in R^r, which means that if any subset S of the corresponding commodities of nodes in R^r share a path from r to node $i \in V$, and then split in node i, then they will never meet again. For example, if $S = \{k_1, k_2, k_3\}$ and in node i the set S splits into $S_1 = \{k_1, k_2\}$ and $S_2 = \{k_3\}$, then k_1 and k_2 will never share an arc again with k_3.

Using these two properties, our formulation is based on two type of decisions, (i) at which nodes we split the subsets of commodities, and (ii) into which subsets of commodities. Note that all the commodities have the same source node, so they all start together. At some point, the set of all commodities will split into different subsets of commodities, and those subsets will also split into other subsets, and so on, until we have each commodity by itself. Our idea is to include a variable that tells us where a subset splits, and the partition that it uses. For instance, say that at some point we have the set $S = \{k_1, k_2, k_3\}$ of commodities that share a path. Eventually, S is split, so the model has to decide where the split is going to happen. In addition, the model has to decide the way that S is split. Note that there are 4 possible partitions of S.

(i) $\{k_1, k_2, k_3\} \rightarrow \{\{k_1, k_2\}, \{k_3\}\}$

(ii) $\{k_1, k_2, k_3\} \rightarrow \{\{k_1, k_3\}, \{k_2\}\}$

(iii) $\{k_1, k_2, k_3\} \rightarrow \{\{k_2, k_3\}, \{k_1\}\}$

(iv) $\{k_1, k_2, k_3\} \rightarrow \{\{k_1\}, \{k_2\}, \{k_3\}\}$

Two observations are, first, we only consider proper partitions, and second, the next partition that we can use will depend on the way that we partitioned S. If we take, for instance, partition (i) then we will have to eventually partition the set $\{k_1, k_2\}$ into $\{k_1\}, \{k_2\}$, but if we take partition (iv) then we do not have to perform any more partitions.

The algorithm proposed in [30] uses a DP algorithm to choose the node where each partition occurs, and the way the sets are partitioned. Because of the latter type decisions, the running complexity of the DP algorithm depends on 3^b. In contrast, we propose to fix the way the sets are partitioned, and just decide the nodes where each partition is performed. To guarantee optimality, we need to solve the problem for every possible way the partitions can occur.

2.2.1 Notation

In this section, we define the notation used in the rest of the chapter. In Section 2.2.2, we provide an example to clarify the concepts introduced in this section.

We define $D = (V, A)$ as a directed graph such that, for every edge $\{i, j\}$ in E we have two arcs (i, j) and (j, i) in A, where V and E are the set of nodes and edges of the original undirected graph G, and where $c_a = c_e$ for all $a \in A$ such that both of its end nodes correspond to the end nodes of $e \in E$. Let $r \in R$ be an arbitrarily selected root node.

We define a set K of commodities, with one commodity for each of the nodes in R^r, and let $b = |K|$. All of the commodities in K share the same source node r, and the sink node of commodity $k \in K$, denoted by t_k, is the corresponding terminal node in R^r. Let S be the set of all subsets of K with at least one element, i.e., $|S| = 2^{|K|} - 1$, and let P be the set of all proper partitions of the elements of S that have cardinality of at least 2.

The underlying r-arborescence of any Steiner tree uses only a subset of the elements of S, which contains the set of all commodities, and all the singletons. This collection of elements of S forms a laminar family. Note that laminar families of sets have been used in solving several combinatorial optimization problems; for further references see [92]. We define $\mathcal{L}_b$ to be the set of admissible laminar families that describe the structure of a Steiner tree with b commodities, where an admissible laminar family is one that contains the set of all commodities, and all the singletons. For laminar family $l \in \mathcal{L}_b$, let $S(l)$ be the set of elements in S that define laminar family l, and let $P(l)$ be the collection of partitions used to split the sets in $S(l)$.

For any $s' \in S(l)$ such that $|s'| \leq |K| - 1$, we say that $\hat{s}$ is its *parent* set, if s' is one of the sets obtained when we partition set $\hat{s}$ according to laminar family l. Note that there is only one parent set for every $s' \in S(l),\ |s'| \leq |K| - 1$. Similarly, for any set $\hat{s} \in S(l)$ such that $|\hat{s}| \geq 2$, we say that s' is a *child* set of $\hat{s}$, if we obtain s' when we partition $\hat{s}$. Note that for every set $\hat{s} \in S(l),\ |\hat{s}| \geq 2$ we have at least 2 child sets. Let $S_l(p)$ be the set of all child nodes of partition p in laminar family l, and let $P_l(s)$ be the partition p that splits s in

laminar family l, which is defined only for sets s, such that $|s| \geq 2$. Finally, we say that a Steiner tree T in G follows an (r, l) *structure*, if the arborescence rooted in r follows the structure defined by laminar family $l \in \mathcal{L}_b$, where $b = |R^r|$.

2.2.2 Example

Suppose we have an instance with 4 terminal nodes. Let r be the root node, and let $K = \{k_1, k_2, k_3\}$. Then,

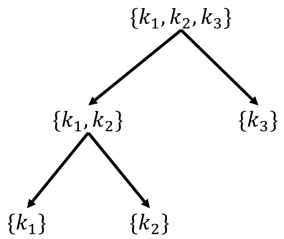
$$S = \left\{ \underbrace{\{k_1\}}_{s_1}; \underbrace{\{k_2\}}_{s_2}; \underbrace{\{k_3\}}_{s_3}; \underbrace{\{k_1, k_2\}}_{s_4}; \underbrace{\{k_1, k_3\}}_{s_5}; \underbrace{\{k_2, k_3\}}_{s_6}; \underbrace{\{k_1, k_2, k_3\}}_{s_7} \right\}$$

and

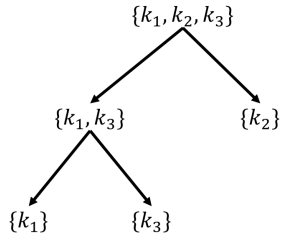
$$P = \left\{ \underbrace{\{(k_1, k_2), (k_3)\}}_{p_1}; \underbrace{\{(k_1, k_3), (k_2)\}}_{p_2}; \underbrace{\{(k_2, k_3), (k_1)\}}_{p_3}; \underbrace{\{(k_1), (k_2), (k_3)\}}_{p_4}; \right.$$
$$\left. \underbrace{\{(k_1), (k_2)\}}_{p_5}; \underbrace{\{(k_1), (k_3)\}}_{p_6}; \underbrace{\{(k_2), (k_3)\}}_{p_7} \right\}$$
$$= \left\{ \underbrace{\{s_4, s_3\}}_{p_1}; \underbrace{\{s_5, s_2\}}_{p_2}; \underbrace{\{s_6, s_1\}}_{p_3}; \underbrace{\{s_1, s_2, s_3\}}_{p_4}; \underbrace{\{s_1, s_2\}}_{p_5}; \underbrace{\{s_1, s_3\}}_{p_6}; \underbrace{\{s_2, s_3\}}_{p_7} \right\}$$

In this case p_1, p_2, p_3 and p_4 are the partitions of the set whose cardinality is 3, which is the set $\{k_1, k_2, k_3\}$. The partitions p_5, p_6 and p_7 are the partitions of sets of cardinality 2, which are the 3 sets $\{k_1, k_2\}$, $\{k_1, k_3\}$ and $\{k_2, k_3\}$.

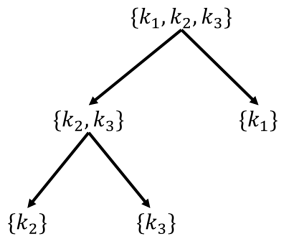
Figure (2.1) shows the tree representation of all the possible laminar families for the case $K = \{k_1, k_2, k_3\}$.



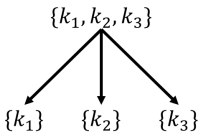
(a) Tree representation of laminar family $l_1 \in \mathcal{L}_3$.



(b) Tree representation of laminar family $l_2 \in \mathcal{L}_3$.



(c) Tree representation of laminar family $l_3 \in \mathcal{L}_3$.



(d) Tree representation of laminar family $l_4 \in \mathcal{L}_3$.

Figure 2.1: Tree representation of all possible laminar families for the case $K = \{k_1, k_2, k_3\}$

In this case, we have

$$P(l_1) = \{p_1, p_5\} \quad S(l_1) = \big\{\{k_1, k_2, k_3\}; \{k_1, k_2\}; \{k_1\}; \{k_2\}; \{k_3\}\big\} \quad = \{s_7, s_4, s_1, s_2, s_3\}$$

$$P(l_2) = \{p_2, p_6\} \quad S(l_2) = \big\{\{k_1, k_2, k_3\}; \{k_1, k_3\}; \{k_1\}; \{k_2\}; \{k_3\}\big\} \quad = \{s_7, s_5, s_1, s_2, s_3\}$$

$$P(l_3) = \{p_3, p_7\} \quad S(l_3) = \big\{\{k_1, k_2, k_3\}; \{k_2, k_3\}; \{k_1\}; \{k_2\}; \{k_3\}\big\} \quad = \{s_7, s_6, s_1, s_2, s_3\}$$

$$P(l_4) = \{p_4\} \qquad S(l_4) = \big\{\{k_1, k_2, k_3\}; \{k_1\}; \{k_2\}; \{k_3\}\big\} \qquad = \{s_7, s_1, s_2, s_3\}$$

Consider the laminar family $l_1 \in \mathcal{L}_3$ shown in Figure (2.1a). We have that $\{k_1, k_2, k_3\}$ is the *parent* set of $\{k_1, k_2\}$ and $\{k_3\}$, and that $\{k_1, k_2\}$ is the *parent* set of $\{k_1\}$ and $\{k_2\}$. On the other hand, sets $\{k_1\}$ and $\{k_2\}$ are the *child* sets of $\{k_1, k_2\}$, and, set $\{k_1, k_2\}$ and $\{k_3\}$

are *child* sets of $\{k_1, k_2, k_3\}$. Finally, we have

$$S_{l_1}(p_1) = \big\{\{k_1, k_2\}; \{k_3\}\big\} = \{s_4, s_3\}$$
$$S_{l_1}(p_5) = \big\{\{k_1\}; \{k_2\}\big\} = \{s_1, s_2\}$$
$$P_{l_1}\big(\{k_1, k_2, k_3\}\big) = P_{l_1}(s_7) = \{p_1\}$$
$$P_{l_1}\big(\{k_1, k_2\}\big) = P_{l_1}(s_4) = \{p_5\}$$

2.3 IP Model

We propose an IP model $\mathcal{Z}_l$ that finds an optimal solution for a given laminar family $l \in \mathcal{L}_b$, where $b = |K|$. Since we are considering a fixed laminar family l, we are given $S(l)$, the set of subsets of K that we use, and the partitions that are performed, as well as the *parent* and *child* sets of each set in $S(l)$. Therefore, the main purpose of our model is to choose the node where each partition is performed. Once we know where each partition is performed, it is easy to determine the arcs that minimize cost, since the problem reduces to finding a shortest path between the nodes where each subset $s \in S(l)$ begins and ends.

For notational simplicity, we do not index the variables of $\mathcal{Z}_l$ by l. Our proposed for-

mulation $\mathcal{Z}_l$ is

$$\min \sum_{a \in A} c_a \left(\sum_{s \in S(l)} f_a^s \right)$$

$$\sum_{a \in \delta^+(i)} f_a^s - \sum_{a \in \delta^-(i)} f_a^s = \hat{y}_i^s - \overline{y}_i^s \qquad \forall i \in V, s \in S(l) \quad (2.1a)$$

$$w_i^p = \overline{y}_i^s \qquad \forall i \in V, s \in S(l) : |s| \geq 2, p = P_l(s) \quad (2.1b)$$

$$w_i^p = \hat{y}_i^s \qquad \forall i \in V, p \in P(l), s \in S_l(p) \quad (2.1c)$$

$$\sum_{i \in V} w_i^p = 1 \qquad \forall p \in P(l) \quad (2.1d)$$

$$\hat{y}_r^K = 1 \quad (2.1e)$$

$$\hat{y}_i^K = 0 \qquad \forall i \in V \backslash \{r\} \quad (2.1f)$$

$$\overline{y}_{t_k}^k = 1 \qquad \forall k \in K \quad (2.1g)$$

$$\overline{y}_i^k = 0 \qquad \forall k \in K, i \in V \backslash \{t_k\} \quad (2.1h)$$

$$f \in \{0,1\}^{|A| \times |S(l)|} \quad (2.1i)$$

$$(\hat{y}, \overline{y}) \in \{0,1\}^{|V| \times |S(l)|} \quad (2.1j)$$

$$w \in \{0,1\}^{|V| \times |P(l)|} \quad (2.1k)$$

The variable f_a^s is 1 if we send flow in arc a for subset s, and 0 otherwise. This means that all of the commodities that compose subset s use arc a. The variable $\hat{y}_i^s$ is 1 if commodities in set s *start* sharing a path in node i but commodities in parent set s' do not, and 0 otherwise. The variable $\overline{y}_i^s$ is 1 if commodities in set s *end* sharing a path in node i, and 0 otherwise. Finally, w_i^p is 1 if a partition p is performed in node i, and 0 otherwise. Constraint (2.1a) is the flow conservation constraints for all the subsets that belong to laminar family l, which relates variables f, $\hat{y}$ and $\overline{y}$. Constraint (2.1b) states that if partition p is performed in node i, then the subset s that is partitioned has to stop sharing in node i. Constraint (2.1c) states that if partition p occurs in node i, then the *child* sets of partition p start to share in node i. Constraint (2.1d) imposes that the partitions can occur only in one

node. Constraints (2.1e) and (2.1f) say that the set of all commodities start sharing in the root node. Finally, constraints (2.1g) and (2.1h) state that every commodity has to send its flow to its sink node.

The LP relaxation of $\mathcal{Z}_l$, which we refer to as $LP(\mathcal{Z}_l)$, is defined by the same objective function and set of linear equalities, but relaxing the integrality condition of the variables. This means that we replace $(2.1i)$, $(2.1j)$ and $(2.1k)$ with

$$f \in [0,1]^{|A|\times|S(l)|} \tag{2.2}$$

$$(\hat{y}, \overline{y}) \in [0,1]^{|V|\times|S(l)|} \tag{2.3}$$

$$w \in [0,1]^{|V|\times|P(l)|} \tag{2.4}$$

For all $l \in \mathcal{L}_b$, let $\mathcal{Q}_l^{IP}$ and $\mathcal{Q}_l^{LP}$ be the feasible regions of $\mathcal{Z}_l$ and $LP(\mathcal{Z}_l)$ respectively. For all $e \in E$, let $A(e) = \{a \in A : a \text{ is a directed arc of } e \in E\}$, and let $\chi \in \{0,1\}^{|E|}$. χ represents a solution to the Steiner tree problem in G, where $\chi_e = 1$ if edge e belongs to the solution.

Definition 1. *Let $l \in \mathcal{L}_b$, and let $x = (f, w, \hat{y}, \overline{y})$ be a feasible solution for $\mathcal{Q}_l^{IP}$. We define ϕ to be a mapping of x to the original problem space, where*

$$\phi(x) = \chi \in \{0,1\}^{|E|}$$

is given by

$$\chi_e = \textit{min}\left\{1, \sum_{s\in S(l)} \sum_{a\in A(e)} f_a^s\right\} \quad \textit{for all } e \in E.$$

2.3.1 Structure of feasible solutions of $\mathcal{Z}_l$

We assume that $\mathcal{Q}_l^{IP}$ is not empty, and let $x = (f, w, \hat{y}, \overline{y})$ be an arbitrary feasible solution for $\mathcal{Z}_l$. A feasible solution can have two components, it must have an acyclic component, denoted by x_{nc}, and it may also have a cycle component, denoted by x_c. Then, $x = x_{nc} + x_c$. The acyclic component is always present and is a feasible solution to the problem, while the cycle component may not be present, i.e., we may have $x_c = 0$. The cycle component is not feasible on its own, since it is only composed of cycles, and therefore it only fulfills the flow constraints (2.1a). We analyze each component separately.

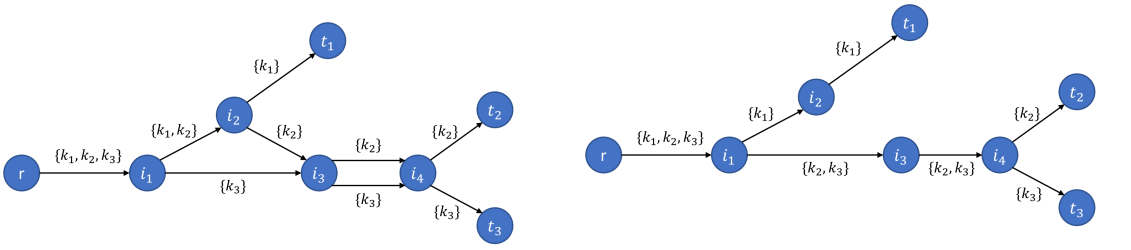
First, consider the acyclic component, $x_{nc} = (f_{nc}, w_{nc}, \hat{y}_{nc}, \overline{y}_{nc})$. Since this component has to be feasible, then by constraints (2.1b)-(2.1h), $\hat{y}_{nc}^s$ and $\overline{y}_{nc}^s$ have exactly one non-zero component for all $s \in S(l)$, which we call i_s and j_s respectively. There are two cases, either $i_s \neq j_s$, or $i_s = j_s$. In the first case, because of constraint (2.1a), there must be a path for s from i_s to j_s. In the second case, since x_{nc} has no cycles, $f_{nc}^s = 0$, or equivalently the set s has no path. On the other hand, constraints (2.1b)-(2.1d) ensure that the path of every parent set ends in the same node where the path of its child sets start. By constraints (2.1e) and (2.1f) we have that the set of all commodities must start in r, and by constraints (2.1g) and (2.1h), each singleton k must end its path in t_k. Therefore, in every feasible solution, there is a path from r to t_k for all k in K, which corresponds to the union of the paths of all sets containing k. Consequently, all Steiner trees with (r, l) structure have a corresponding acyclic feasible solution to $\mathcal{Z}_l$, but not all acyclic solutions of $\mathcal{Z}_l$ map to an (r, l) structured Steiner tree. We will discuss the case in which a feasible acyclic solution is not an (r, l) structured Steiner tree in Proposition 1.

Now, consider the cycle component. Let $x_c = (f_c, w_c, \hat{y}_c, \overline{y}_c) \neq 0$, i.e, there is at least one set $\tilde{s} \in S(l)$ which has at least one cycle, and for simplicity of the argument, suppose that $\tilde{s}$ has only one cycle. The only way that this can happen is when the cycle uses arcs that are not in the path between $i_{\tilde{s}}$ and $j_{\tilde{s}}$, the path used by $\tilde{s}$ in the non-cycle component, otherwise it would be an infeasible solution because it would violate constraints (2.1a) and

(2.1i). In the cycle component, w_c, $\hat{y}_c$ and $\tilde{y}_c$ are always 0, and if $x_c \neq 0$ then $f_c \neq 0$. Consequently, we have $w = w_{nc}$, $\hat{y} = \hat{y}_{nc}$, $\tilde{y} = \tilde{y}_{nc}$, and $f = f_{nc} + f_c$. If a solution does not contain cycles, then $f = f_{nc}$.

Proposition 1. *For any $l \in \mathcal{L}_b$, let x be a feasible solution of $\mathcal{Z}_l$ that does not contain any cycles. Then $\phi(x)$ is either an (r, l) structured Steiner tree, or the support of $\phi(x)$ contains a Steiner tree with a different (r, l) structure.*

Proof. As was pointed out in Section 2.3.1, in the non-cycle component of every feasible solution x of $\mathcal{Z}_l$, there is a path from r to t_k for every commodity $k \in K$. This is a property of every Steiner tree with an (r, l) structure. But, there are also other cases where a feasible solution x of $\mathcal{Z}_l$ does not map to an (r, l) structured Steiner tree, as shown in Figure 2.2. We observe in Figure 2.2a, a feasible solution x of $\mathcal{Z}_{l_1}$ whose mapping $\phi(x)$ is not an (r, l_1) structured Steiner tree[1]. Note that commodities k_2 and k_3 share arcs, but not using set $\{k_2, k_3\}$ because this set is not in l_1.



(a) Feasible solution x for $\mathcal{Z}_{l_1}$.

(b) Steiner tree with (r, l_3) structure, contained in support of $\phi(x)$.

Figure 2.2: Feasible solution x of $\mathcal{Z}_{l_1}$, whose mapping $\phi(x)$ is not an (r, l_1) structured Steiner tree, and (r, l_3) structured Steiner tree contained in support of $\phi(x)$.

Now, suppose $x \in \mathcal{Q}_l^{IP}$ is acyclic and does not map to an (r, l) structured Steiner tree. The mapping $\phi(x)$ tells us which edges of the original undirected graph G are used by x. We can construct a subgraph defined by those edges, and we know there must be a Steiner tree in it, because in solution x there is at least one path from r to t_k for all $k \in K$. Then, we can take any Steiner tree within the constructed subgraph, which will have a unique (r, l^*)

[1] See Figure 2.1 for a description of laminar families l_1 and l_3.

structure, with l^* not necessarily equal to l. We can do this by taking any spanning tree in the constructed subgraph. For instance, Figure 2.2b shows an (r, l_3) structured Steiner tree within the support of $\phi(x)$ shown in Figure 2.2a. □

2.3.2 Structure of feasible solutions of $LP(\mathcal{Z}_l)$

The structure of the feasible solutions of $LP(\mathcal{Z}_l)$ is similar to the solutions of $\mathcal{Z}_l$, because they also have a non-cycle, and a cycle component. For simplicity, we use the same notation as in Section 2.3.1. We assume that $\mathcal{Q}_l^{LP}$ is not empty, and let $x = (f, w, \hat{y}, \tilde{y})$ be an arbitrary solution of $LP(\mathcal{Z}_l)$. Let $x = x_{nc} + x_c$, where x_{nc} is the non-cycle component of x, and x_c is the cycle component of x.

The analysis for this case is similar to the analysis in Section 2.3.1, but since the variables now can be fractional, we may have solutions where the non-cycle and the cycle component use the same arc a for the same set s, but the sum of acyclic and cyclic components has to be at most 1 for all arcs and for all sets. Moreover, in the cycle component, we may have that two different cycles, for the same s, use the same arc a. Thus, the cycle part will be a collection of weighted cycles for each $s \in S(l)$, such that for every arc $a \in A$, the sum of acylic and cyclic flow in a for s is at most 1.

Let $LP(\mathcal{Z}_l)(\lambda)$ denote the formulation $LP(\mathcal{Z}_l)$, when we replace 1 by $\lambda \in (0, 1]$ in constraints (2.1d), (2.1e), (2.1g), (2.2), (2.3), and (2.4). Let $OPT(\lambda)$ denote the value of an optimal solution to $LP(\mathcal{Z}_l)(\lambda)$, for $\lambda \in (0, 1]$.

Lemma 1. *For every $l \in \mathcal{L}_b$ and $\lambda \in (0, 1]$, we have $OPT(\lambda) = \lambda\,[OPT(1)]$.*

Proof. For any $\lambda \in (0, 1]$, let $(f^\lambda, \hat{y}^\lambda, \overline{y}^\lambda, w^\lambda)$ denote an optimal solution for $LP(\mathcal{Z}_l)(\lambda)$. Then,

- $OPT(\lambda) \leq \lambda\,[OPT(1)]$. Since $(\lambda f^1, \lambda \hat{y}^1, \lambda \overline{y}^1, \lambda w^1)$ is a feasible solution to $LP(\mathcal{Z}_l)(\lambda)$, it holds that $OPT(\lambda) \leq \lambda\,[OPT(1)]$.

- $OPT(\lambda) \geq \lambda\,[OPT(1)]$. Since $(\frac{1}{\lambda}f^{\lambda}, \frac{1}{\lambda}\hat{y}^{\lambda}, \frac{1}{\lambda}\overline{y}^{\lambda}, \frac{1}{\lambda}w^{\lambda})$ is a feasible solution to $LP(\mathcal{Z}_l)(1)$, it holds that $\frac{1}{\lambda}\,[OPT(\lambda)] \geq OPT(1)$

Then, $OPT(\lambda) = \lambda\,[OPT(1)]$ for all $\lambda \in (0, 1]$. □

Our main result is stated in the following theorem.

Theorem 1. *For any $l \in \mathcal{L}_b$, we have Conv$(\mathcal{Q}_l^{IP}) = \mathcal{Q}_l^{LP}$.*

Proof. We will prove that for all cost vectors c, there exists an optimal solution to $LP(\mathcal{Z}_l)$ that is integral.

The following definitions will be used in the proof. For all $i \in V, s \in S(l)$, let $LP(\mathcal{Z}_l(i,s))$ be a subformulation of $LP(\mathcal{Z}_l)$, where the root is i (instead of r), the terminal nodes are t_k for all $k \in s$, and the splitting sequence is defined by the way s, and its subsets, are split in laminar family l. The cost vector of $LP(\mathcal{Z}_l(i,s))$ is the same as the one used in $LP(\mathcal{Z}_l)$, but only considering the components of $LP(\mathcal{Z}_l)$ that are present in $LP(\mathcal{Z}_l(i,s))$. Let $x^*(i,s)$ be an optimal solution to $LP(\mathcal{Z}_l(i,s))$.

Let $x^* = (f^*, \hat{y}^*, \overline{y}^*, w^*)$ be an optimal solution of $LP(\mathcal{Z}_l)$. Note that if w^* is integer, then $\hat{y}^*$ and $\overline{y}^*$ are integer because of constraints (2.1b)-(2.1c). This implies that for all $s \in S(l)$, there is only one non-zero component in vectors $\hat{y}^s$ and $\overline{y}^s$, which is equal to 1. Let i_s and j_s be those indexes, respectively. Now, for all $s \in S(l)$, constraints (2.1a) and (2.1i) correspond to the shortest i_s-j_s path polytope, which has only integer extreme points. Since each of these polytopes is independent of each other, we have that f^* is integer. Therefore, if w^* is integer, then x^* is integer.

Now, suppose that w^* is fractional. This implies that for at least one set $s \in S(l)$, $\overline{y}^s$ is fractional. Note that we may have that $\hat{y}^s$ is also fractional, but for at least one $s \in S(l)$, we claim that $\hat{y}^s$ is integer, and $\overline{y}^s$ is fractional. The justification for this claim is the following. Suppose the partitions in $P(l)$ are ordered in decreasing order by the cardinality of the set they split, i.e., the first element of $P(l)$ splits K, and the last set of $P(l)$ splits a set

of cardinality 2. Now, consider w^*, and let $\dot{p}$ be the first element in $P(l)$ such that $w^{\dot{p}}$ is fractional. Let $\dot{s}$ be the set that $\dot{p}$ splits. Then, it is clear that $\hat{y}^{\dot{s}}$ is integer and $\overline{y}^{\dot{s}}$ is fractional.

Let $\hat{s}$ be the largest set in $S(l)$ such that $\hat{y}^{\hat{s}}$ is integral, and $\overline{y}^{\hat{s}}$ is fractional, and for simplicity of the argument, suppose $\hat{s} = K$. Let $I(\hat{s})$ be the set of indexes such that $\overline{y}_i^{\hat{s}} = \lambda_i > 0$ for all $i \in I(\hat{s})$, and $\sum_{i \in I(\hat{s})} \lambda_i = 1$. Let $\hat{p} \in P(l)$ be the partition that splits $\hat{s}$ in laminar family l. By constraint (2.1b) we have $w_i^{\hat{p}} = \lambda_i$ for all $i \in I(\hat{s})$, as shown in Figure 2.3.

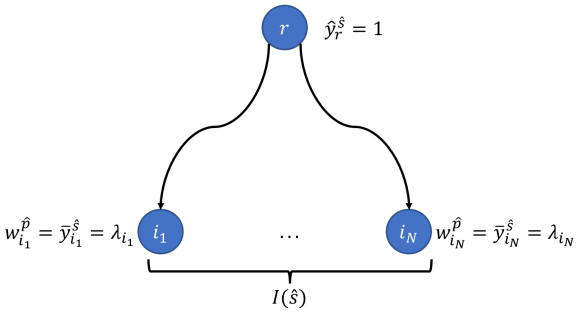


Figure 2.3: Solution illustration.

For all $i \in I(\hat{s})$ and $s \in S_l(\hat{p})$, by constraint (2.1c), we have $\hat{y}_i^s = \lambda_i$. Now, take a fixed $s \in S_l(\hat{p})$, and a fixed $i \in I(\hat{s})$. Because $\hat{y}_i^s = \lambda_i$, then we know that within solution x^*, we are sending λ_i units of flow from node i to $\{t_k\}_{k \in s}$. Since x^* is optimal, then by Lemma 1, that part of the solution has to correspond to $\lambda_i x^*(i, s)$, otherwise, we can improve solution x^*. On the other hand, within solution x^* we are sending λ_i units of flow from r to i, for all $i \in I(\hat{s})$. Let $x^*(r, i, \hat{s})$ be the lowest cost solution sending 1 unit of flow from r to $i \in I(\hat{s})$ for set $\hat{s}$, then, by Lemma 1, $\lambda_i x^*(r, i, \hat{s})$ corresponds to the lowest cost solution sending λ_i units of flow from r to $i \in I(\hat{s})$ for set $\hat{s}$, otherwise we can improve solution x^*. Figure 2.4 shows a representation of the last statement.

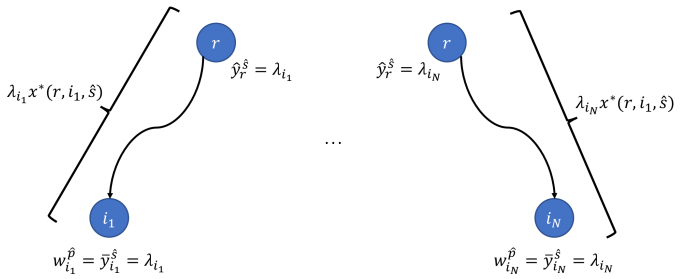


Figure 2.4: Solution decomposition illustration.

Putting everything together, we have

$$x^* = \sum_{i \in I(\hat{s})} \lambda_i \left[x^*(r, i, \hat{s}) + \sum_{s \in S_l(\hat{p})} x^*(i, s) \right]$$

But, for all $i \in I(\hat{s})$, $x^*(r, i, \hat{s}) + \sum_{s \in S_l(\hat{p})} x^*(i, s)$ is a feasible solution. Then, x^* is a convex combination of $\{x^*(r, i, \hat{s}) + \sum_{s \in S_l(\hat{p})} x^*(i, s)\}_{i \in I(\hat{s})}$, and therefore, it is not an extreme point. Note that, $x^*(r, i, \hat{s})$ and $\sum_{s \in S_l(\hat{p})} x^*(i, s)$ may contain cycles. Furthermore, if $\hat{s} \neq K$, we can use the same argument, but we have to consider the solution of all the sets that contain $\hat{s}$ as a subset. By definition of $\hat{s}$, in the solution of all those sets, $\hat{y}$ and $\overline{y}$ will be integers, and therefore, the entire solution of all those sets has to be integer. Thus, we can include that part of the solution in $x^*(r, i, \hat{s}) + \sum_{s \in S_l(\hat{p})} x^*(i, s)$ for all $i \in I(\hat{s})$, and use the same argument as before. Consequently, all extreme point solutions have $\hat{y}, \overline{y}$, and w integral, and therefore, all extreme points of $\mathcal{Q}_l^{LP}$ are integral. □

As a corollary, we can solve the entire Steiner tree problem by optimizing each laminar family subproblem independently, and taking the solution with the lowest cost.

Corollary 1. *Let $c > 0$ be the cost vector of a Steiner tree problem instance. Let $x^l \in \operatorname{argmin}\{c^T x : x \in \mathcal{Q}_l^{LP}\}$, and $v^l = c^T x^l$ for all $l \in \mathcal{L}_b$. Then, $\phi(x^{l^*})$ is a minimum cost Steiner tree, where $l^* \in \operatorname{argmin}\{v^l : l \in \mathcal{L}_b\}$.*

Proof. Note that, since the cost is positive, then the optimal solution for each subproblem is acyclic. By Proposition 1, we know that $\phi(x^{l^*})$ is either an (r, l^*) structured Steiner tree,

or $\phi(x^{l^*})$ contains a Steiner tree, with a different structure. But, since the cost vector is non-negative, then $\phi(x^{l^*})$ has to be a Steiner tree, otherwise, we can take any Steiner tree contained in the support of $\phi(x^{l^*})$, which will have a lower cost since it uses a subset of the edges used by $\phi(x^{l^*})$. Therefore, $\phi(x^{l^*})$ is a minimum cost Steiner tree. □

Note that using a standard disjunctive programming argument, we can write an extended formulation for the entire problem based on the formulations for each laminar family [4, 21].

An important remark is that, although we have done all the analysis to solve the undirected Steiner tree problem, this approach can also be used to solve the directed Steiner tree problem, which is a more difficult problem to solve. For the directed version of the problem, the main difference is that we are given the directed graph $D = (V, A)$ and the root node r. The rest of the analysis remains the same.

2.3.3 Problem size

The downside of our proposed formulation is the number of laminar families, which grows very fast as the number of terminals increases. In this section we provide an upper bound on the number of laminar families that we have to consider, and we show that this upper bound is tight.

In the following proposition, we use the notion of a *full binary* tree. Note that a tree T is called a *full binary* tree if every node in T has either zero or two children.

Proposition 2. *To solve the Steiner tree problem to optimality, it suffices to consider only laminar families in $\mathcal{L}_b$ whose tree representation corresponds to a full binary tree.*

Proof. Let l_1 be a laminar family in $\mathcal{L}_b$ and let $T(l_1)$ be the tree representation of l_1, such that $T(l_1)$ is not a full binary tree. Therefore, there must be a node in $T(l_1)$ that has at least 3 child nodes, which implies that there must be a set in l_1 that is partitioned into at least 3 sets; let $\hat{s}$ be such set. Suppose that the *child* sets of $\hat{s}$ in l_1 are $s_1, s_2, \ldots, s_t$ with $t \geq 3$. For

simplicity, assume $t = 3$. Now, let l_2 be a laminar family in $\mathcal{L}_b$ such that $S(l_2) = S(l_1) \cup s'$ where $s' = s_1 \cup s_2$. This means that all partitions in l_1 and l_2 are the same, but in l_1, $\hat{s}$ is partitioned into s_1, s_2, s_3, and in l_2, $\hat{s}$ is partitioned into s' and s_3, and s' is partitioned into s_1 and s_2. Now, let $x_1 = (f_1, w_1, \hat{y}_1, \overline{y}_1)$ be an optimal solution to $\mathcal{Z}_{l_1}$. We claim that we can construct a feasible solution to $\mathcal{Z}_{l_2}$ using x_1. Let i^* be the node where $\hat{s}$ is split in x_1. Then we define $x_2 = (f_2, w_2, \hat{y}_2, \overline{y}_2) \in \mathcal{Q}_{l_2}^{LP}$ as follows

$$(f_2)_a^s = \begin{cases} (f_1)_a^s & \forall a \in A, s \in S(l_1) \\ 0 & \forall a \in A, s = s' \end{cases}$$

$$(\hat{y}_2)_i^s = \begin{cases} (\hat{y}_1)_i^s & \forall i \in V, s \in S(l_1) \\ 1 & i = i^*, s = s' \\ 0 & \forall i \in V \backslash \{i^*\}, s = s' \end{cases}$$

$$(\overline{y}_2)_i^s = \begin{cases} (\overline{y}_1)_i^s & \forall i \in V, s \in S(l_1) \\ 1 & i = i^*, s = s' \\ 0 & \forall i \in V \backslash \{i^*\}, s = s' \end{cases}$$

Note that the values of the w_2 vector will be determined by the values of $\hat{y}_2$ and $\overline{y}_2$. Moreover, x_2 is a feasible solution to $\mathcal{Z}_{l_2}$, whose cost is the same as the cost of x_1. Hence, $\mathcal{Z}_{l_2} \leq \mathcal{Z}_{l_1}$. The same argument holds for $t > 3$ as well. Consequently, we only need laminar families where each non-singleton set is partitioned into two proper subsets, which correspond to laminar families whose tree representation is a full binary tree. □

Using Proposition 2, we can reduce the number of laminar families by considering only laminar families whose tree representations are full binary trees. For simplicity of notation, we denote by $\mathcal{L}_b$ the reduced set of laminar families, when we have $b+1$ terminals. We are interested in expressing $|\mathcal{L}_b|$ as a function of b. Note that $|\mathcal{L}_b|$ is equivalent to the number

of full binary labeled trees with b leaves. It is known that (see [9], and [99] Chapter 5.2.6)

$$|\mathcal{L}_b| = (2b-3)!! = \frac{(2(b-1))!}{2^{b-1}(b-1)!}$$

Note, however, that $|\mathcal{L}_b|$ is not always achieved, since there are graphs that do not yield some laminar families of $\mathcal{L}_b$, which is why $\frac{(2(b-1))!}{2^{b-1}(b-1)!}$ is an upper bound on the number of laminar families to consider. Now, recall Stirling's formula, which gives lower and upper bounds for $n!$. For any $n \in \mathbb{Z}^+$ we have

$$\sqrt{2\pi} n^{n+\frac{1}{2}} e^{-n} \leq n! \leq e n^{n+\frac{1}{2}} e^{-n}$$

Then,

$$\begin{aligned}
|\mathcal{L}_b| &= \frac{(2(b-1))!}{2^{b-1}(b-1)!} \\
&\leq \frac{e(2(b-1))^{2(b-1)+\frac{1}{2}} e^{-2(b-1)}}{2^{b-1}\sqrt{2\pi}(b-1)^{(b-1)+\frac{1}{2}} e^{-(b-1)}} \\
&= \frac{e 2^{(b-1)} (b-1)^{(b-1)} e^{-(b-1)}}{\sqrt{\pi}} \\
&= \frac{e}{\sqrt{\pi}} \left(\frac{2(b-1)}{e} \right)^{(b-1)}
\end{aligned}$$

We conclude that $|\mathcal{L}_b|$ grows as $\mathcal{O}\left(\left(\frac{2b}{e}\right)^b\right)$. Although this is a big number, it is small compared to the total number of laminar families. In fact, it can be shown that as b increases, the ratio between the number of laminar families corresponding to full binary trees and the total number of admissible laminar families, goes to 0.

Now, we analyze the size of each subproblem $LP(\mathcal{Z}_l)$ for $l \in \mathcal{L}_b$. By Proposition 2, we use laminar families whose tree representation $T(l)$ is a full binary tree. An important property of full binary trees is that the number of internal nodes is $L-1$, where L is the number of leaves. In our case, there are b leaves, which implies $b-1$ internal nodes, and $2b-1$ nodes in the entire tree. This implies that $|S(l)| = 2b-1$ for all $l \in \mathcal{L}_b$. Moreover,

the number of non-singleton sets corresponds to the number of internal nodes in $T(l)$ which is $b-1$, and also is the number of partitions in l. Finally, since we only consider l such that $T(l)$ is a full binary tree, the number of *child* sets of each partition is exactly 2. Using these facts, the number of variables and constraints are the same for all $l \in \mathcal{L}_b$, and the number of variables is $\mathcal{O}(nb + mb)$ and the number of constraints is $\mathcal{O}(nb + b)$, where $n = |V|$, $m = |A|$ and $b + 1 = |R|$. Therefore, $LP(\mathcal{Z}_l)$ is of polynomial size in the input data. Consequently, we have that each subproblem is polynomially solvable, but the number of subproblems grows as $\mathcal{O}\left(\left(\frac{2b}{e}\right)^b\right)$, which implies that the problem is fixed-parameter tractable with respect to the number of terminal nodes. This last remark is consistent with previous results [30, 40].

2.4 Computational Results

We provide computational results to evaluate the performance of our proposed formulation. These results are presented just to validate our model. At the current state of the proposed approach, the results are not competitive with the state of the art solvers for the unidirected and directed Steiner tree problems [32, 52, 43, 58, 3, 83, 82, 107].

We use instances from the SteinLib library [59]. Table 2.1 shows the number of laminar families as a function of the number of terminal nodes, and it shows how long it will take to solve the entire problem if each subproblem takes one second to execute, and they are solved sequentially.

Table 2.1: Number of terminals, maximum number of laminar families, and total execution time if each subproblem takes 1 second to solve.

Number of terminals	$\lvert\mathcal{L}_b\rvert$	Running time
3	1	1 sec
4	3	3 sec
5	15	15 sec
6	105	1.75 min
7	945	15.75 min
8	10,395	2.89 hrs
9	135,135	37.54 hrs
10	2,027,025	23.46 days
11	34,459,425	1.09 yrs

As we can see from Table 2.1, the expected time to run instances with 9 or more terminals will be very large, even if the execution time for each subproblem is 1 second. Thus, we have only run experiments on instances with at most 8 terminal nodes.

Our code is implemented in *Python* using *Gurobi* 7.5.1 as a solver. All tests were performed on a laptop with an *Intel Core i7* (3.3 GHz) processor, and 16 GB of memory, using the *macOS Sierra* operating system. As we previously discussed, we can solve each laminar family subproblem independently, and therefore, they can be solved in parallel, but we implemented our code such that we run each subproblem sequentially.

Table 2.2 summarizes the results. The first two columns correspond to the test set and instance name, the third column shows the number of nodes, the fourth column shows the number of arcs, the fifth and the sixth columns present the number of terminal nodes and total number of laminar families, respectively. Finally, the last two columns present the average execution time for each subproblem, and the total execution time for all the

models, respectively. All the reported times are in seconds.

Table 2.2: Instance details and execution times in seconds.

Test set	Instance	$\|V\|$	$\|A\|$	$\|R\|$	$\|\mathcal{L}_b\|$	Subproblem time	Entire problem time
LIN	*lin01*	53	160	4	3	0.003 s	0.01 s
LIN	*lin02*	55	164	6	105	0.005 s	0.48 s
LIN	*lin04*	157	532	6	105	0.021 s	2.18 s
LIN	*lin07*	307	1,052	6	105	0.060 s	6.26 s
I160	*i160-033*	160	640	7	945	0.016 s	15.43 s
I160	*i160-043*	160	5,088	7	945	0.151 s	142.63 s
I160	*i160-045*	160	5,088	7	945	0.164 s	154.74 s
LIN	*lin03*	57	168	8	10,395	0.007 s	71.26 s
PUC	*cc3-4p*	64	576	8	10,395	0.018 s	190.89 s
PUC	*cc3-4u*	64	576	8	10,395	0.019 s	192.65 s
I320	*i320-011*	320	3,690	8	10,395	0.197 s	1,806.00 s
I320	*i320-043*	320	20,416	8	10,395	0.910 s	9,468.00 s

In order to show the potential of the proposed approach, we present results on larger instances, which are not meant to be solved to optimality, but to obtain a good solution within seconds. As discussed, the main drawback of the proposed approach is that, in order to prove optimality, one must solve the entire set of tree structures. Nevertheless, we may want to solve the problem for a restricted number of structures, and take the best solution among them. This process will not guarantee optimality, but may be useful to have good quality solutions within seconds. The idea is to determine good candidate structures to be solved. There are many ways to do this, and some ideas are discussed in Section 2.5, but we decided to propose a simple way to solve the problem for only one candidate structure, and compare its value with the value of an optimal solution.

The process to construct the structure to solve is the following. First, we arbitrarily

select a root node r. Secondly, we create a complete graph G' whose nodes are the terminal nodes excluding r. For every edge $e' = (t', t'')$, its cost is the length of the shortest path between t' and t'' in the original graph G. Then, we construct a minimum spanning tree T' in G'. Finally, we run Algorithm 1 to construct a laminar family using T' as input.

Algorithm 1 Algorithm to create candidate laminar family l.

Require: Tree T', we assume T' is a set of edges ordered in increasing order of their lengths

Set $S(l) \leftarrow \{\{k_1\}, \{k_2\}, \ldots, \{k_b\}\}$

for $e' \in T'$ **do**

    Let $e' = (t', t'')$, and let k' and k'' be the commodities of terminals t' and t'', respectively.

    Let s' be largest set in $S(l)$ containing k'

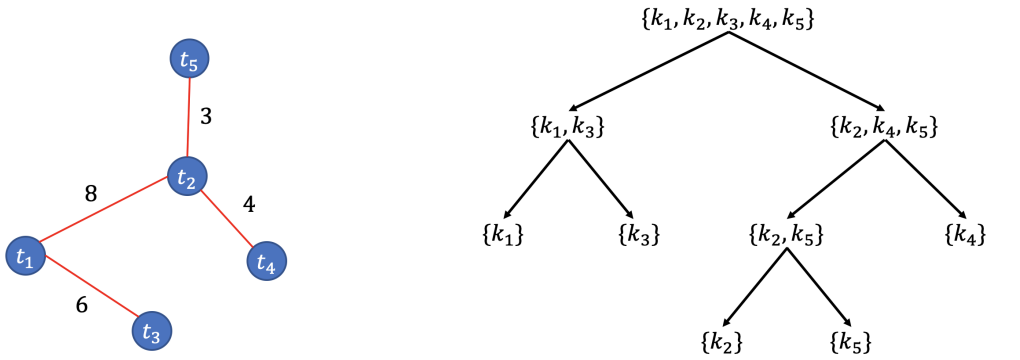
    Let s'' be largest set in $S(l)$ containing k''

    $\hat{s} \leftarrow s' \cup s''$

    $S(l) \leftarrow S(l) \cup \hat{s}$

return $S(l)$.

Algorithm 1 constructs a laminar family l based on the single linkage clustering algorithm. It first starts with the collection $S(l)$ of all the singletons. Then, it goes over the set of all edges of T', which are ordered in increasing order of their lengths. For each edge $e' = (t', t'')$, we take the commodities k' and k'' of terminals t' and t'', respectively. Then, we take sets s' and s'', which are the largest sets in $S(l)$ that contain k' and k'', respectively. Finally, we create a set $\hat{s}$ to be the union of s' and s'', and we add it to $S(l)$. After we visit all edges, we have that $S(l)$ is a collection of sets that forms an admissible laminar family. Figure 2.5 shows an example to understand the laminar construction algorithm. Figure 2.5a shows tree T' in G', and Figure 2.5b shows the tree representation of the laminar family constructed using Algorithm 1.



(a) Minimum spanning tree T' of graph G'.

(b) Tree representation of laminar family constructed from T'.

Figure 2.5: Explanatory example of Algorithm 1.

We take 4 set of instances from the SteinLib that are difficult to solve since they are built to defy preprocessing. The instances chosen were I080, I160, I320, and I640. We only take instances with at least 17 terminal nodes whose optimal value is known. In total, we solve 220 instances. Figure 2.6 shows the performance profile of the proposed approach for each one of the set of instances. We show the performance profile of each set separately, and also show the performance profile of all 4 set of instances, which we refer as *iSeries* in Figure 2.6. To explain the graph, take, for instance, the point $(10, 0.36)$ of the red curve. This means that for the test set I080, 36% of the instances have an error of at most 10%.

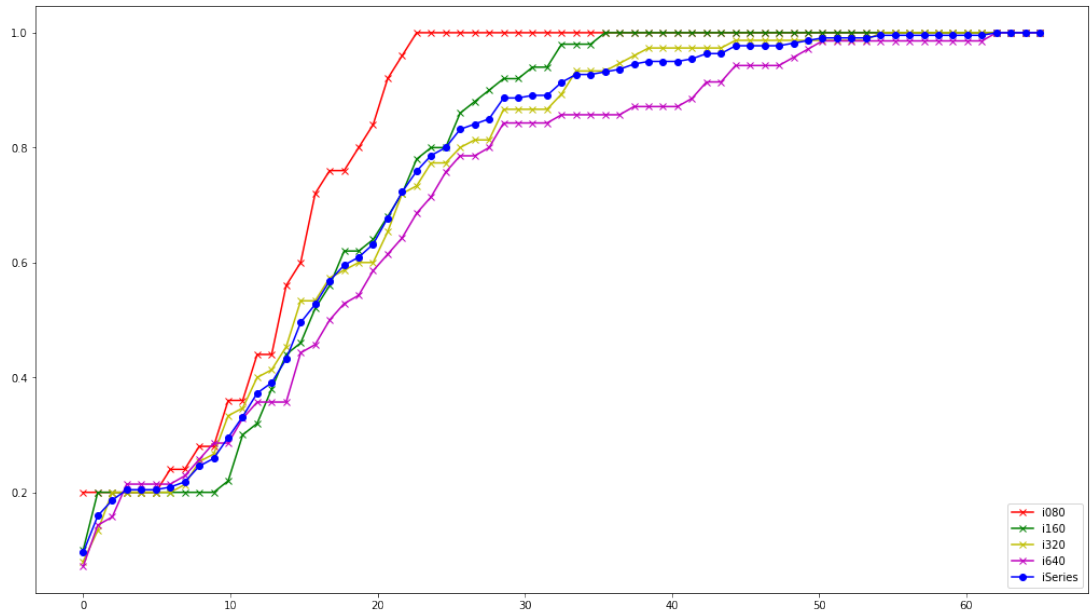


Figure 2.6: Performance profile for studied instances

From Figure 2.6, we see that the solution quality of the proposed approach is fairly good, considering that we only solve the problem for one laminar family. If we consider all the instances studied, 80% of those instances are within a factor of 1.25 of the optimum value, and all of the instances are within a factor of 1.62 of the optimum value.

This approach could be improved, but we wanted to show the potential of our approach to construct good quality solutions. Note that all of the instances solved with this approach took less than 5 seconds to solve.

2.5 Conclusions and Future Work

We propose an LP based approach to the Steiner tree problem. The proposed approach consists of solving a set of independent IPs. The best solution among the set of IPs corresponds to an optimal Steiner tree. Each IP is polynomial in the size of the underlying graph, and we prove that the LP relaxation of each IP is integral, so each IP can be solved as a linear program. The main issue is that the number of IPs to solve grows as $\mathcal{O}\left(\left(\frac{2b}{e}\right)^b\right)$ where $b+1$ is the number of terminal nodes. Consequently, we are able to solve the Steiner

tree problem by solving a polynomial number of LPs, when the number of terminals is fixed. This is consistent with previous results [30, 40].

In Chapter 3 we present an algorithm to solve each subproblem efficiently. The structure of the optimal solution for each problem is well characterized. For each subproblem, the problem reduces to finding the splitting nodes, and then taking the union of shortest paths. Furthermore, we use this algorithm to construct a local search algorithm. Suppose we have a solution for a given laminar family $l \in \mathcal{L}_b$. There are laminar families in $\mathcal{L}_b$ that have a similar topology to that of l.

Future research might be directed in trying to develop tools to deal with the large number of laminar families that need to be considered. An idea is to use column generation to solve this problem. We can start with a small set of the laminar families, and then try to add new laminar families to the problem in a smart way. The main difficulty with this approach is to develop an efficient pricing problem to determine which laminar families must be added to the problem.

Finally, a different line of research is to study our proposed approach in special graphs. There are important results in the literature for special graphs, in particular there are results that provide an upper bound on the integrality gap for well-known formulations. In these type of graphs, we may find a bound on how bad the optimal solution of the subproblem is, for the "worst" structure compared with the optimal solution for the entire problem. Moreover, there are results that show that for special cases of planar graphs, there is an algorithm to solve the problem in polynomial time [5]. These results may imply that for those type of graphs, the number of tree structures to consider is limited.

CHAPTER 3
A SIMULATED ANNEALING ALGORITHM FOR THE DIRECTED STEINER TREE PROBLEM

3.1 Introduction

In this chapter, we focus on the approach proposed in Chapter 2. The main drawback of the proposed approach presented in Chapter 2 is that the number of IPs to solve, grows exponentially with the number of terminal nodes. To address this issue, in this chapter we present a simulated annealing based framework to solve the problem. We focus on the directed setting of the Steiner tree problem, which is known to be much harder computationally.

3.1.1 Our contributions

Our main contributions are the following.

1. We propose a dynamic programming (DP) algorithm to solve each IP efficiently. Computational results show that the proposed algorithm is much faster than solving the LP relaxation of each IP using commercial solvers as Gurobi or CPLEX.

2. We provide a complete characterization of the neighborhood of every tree structure, by describing the neighborhood of its corresponding laminar family.

3. The number of IPs grows exponentially with the number of terminals in the Steiner tree. To address the latter issue, we propose a local-search based algorithm to solve the problem for big instances. We compare our proposed local-search approach with state-of-the-art algorithms to solve the directed version of the Steiner tree problem. The proposed approach outperforms such algorithms in solution quality, while solv-

ing the problem in a few seconds for most of the studied instances, and a few minutes for larger instances.

The remainder of the chapter is organized as follows. Section 3.2 presents an efficient algorithm to solve each sub-problem of the approach presented in Chapter 2. Section 3.3 provides a complete characterization of the neighborhood of each sub-problem, which is needed since simulated annealing performs a local search at each iteration. Section 3.4 presents the proposed simulated annealing framework to solve the problem. This section also provides a routine to check and, potentially, improve the quality of the solutions obtained at each iteration of the simulated annealing algorithm. Moreover, this section also presents a special analysis for the case of rectilinear graphs. In Section 3.5, we present computational experiments obtained by solving the directed Steiner tree problem using our proposed formulation. Section 3.6 gives conclusions.

3.1.2 Notation

The notation used in this chapter is the sames as the used in Chapter 2, but for this chapter we assume we have a directed Steiner tree instance. Let $D = (V, A)$ be the directed input graph, where V is the set of nodes of the graph, and A is the set of arcs of the graph. We are also given a root node $r \in V$, and a set of terminal nodes $R \subseteq V \backslash \{r\}$.

3.2 Algorithm to solve $\mathcal{Z}_l$

3.2.1 Observations and algorithm intuition

Consider a fixed laminar family $l \in \mathcal{L}_b$, and let $s \in S(l)$ such that $2 \leq |s| < |K|$. Since we are in a particular laminar family l, then we know the way s, and its subsets, split. Now, suppose we define that the path of set s starts at node i, then the solution is equivalent to finding a minimum Steiner tree rooted at i, with $|s|$ commodities[1], that follows the splitting

[1]This is equivalent to a directed Steiner tree whose root node is i, and that has $|s|$ terminal nodes which correspond to $\{t_k\}_{k \in s}$

sequence for s defined in l. We will denote $l(s)$, to the "sub-laminar" family of l, when we only focus on s.

As mentioned in Chapter 2, for every laminar family, the problem reduces to finding the splitting nodes, because once we have the splitting nodes, then we only need to connect the corresponding nodes by the shortest paths between them. The proposed algorithm uses this fact, and since we do not know the splitting nodes in advance, then we need to compute the shortest path between all pair of nodes in D.

Consider the following example. Suppose we want to solve a directed Steiner tree with 4 terminals. Also, suppose we are solving the sub-problem for the laminar family shown in Figure 3.1

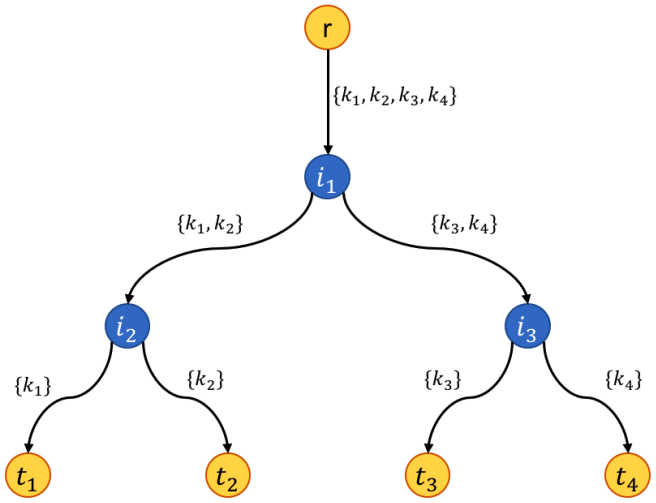


Figure 3.1: Laminar family representation.

We have to find nodes i_1, i_2, and i_3, shown in Figure 3.1. The idea of the proposed algorithm to solve $\mathcal{Z}_l$ is the following. For all sets $s \in S(l)$ with $2 \leq |s| < |K|$, and for all node $i \in V$, we define the minimum $(i, l(s))$ structured Steiner tree, whose optimal solution will be denoted by $x(i, s)$. Now, consider the set $\{k_1, k_2\}$ in the example shown in Figure 3.1. If we fix a given node i as a root for its corresponding Steiner tree, then we can compute $x(i, \{k_1, k_2\})$ in the following fashion. For all $j \in N$, we compute the

sum of 3 shortest paths, which are an i-j shortest path, a j-t_1 shortest path, and a j-t_2 shortest path. Let j^* be a node such that the sum is the smallest, then $x(i, s)$ is the union of an i-j^* shortest path, a j^*-t_1 shortest path, and a j^*-t_2 shortest path. We repeat that procedure for all $i \in V$ and we will have computed all possible Steiner trees for $\{k_1, k_2\}$. This procedure is the equivalent for $\{k_3, k_4\}$. For set $\{k_1, k_2, k_3, k_4\}$, the process is similar, since $x(i, \{k_1, k_2, k_3, k_4\})$ is going to be the union of 3 solutions, but in this case, it is the union of a shortest i-j^* path, $x(j^*, \{k_1, k_2\})$ and $x(j^*, \{k_3, k_4\})$. Note, that since in this case $K = \{k_1, k_2, k_3, k_4\}$, then we only care about $x(r, \{k_1, k_2, k_3, k_4\})$.

3.2.2 Proposed algorithm

Before stating the algorithm, let us introduce the notation used within this section.

- $sp(i, j)$: Collection of arcs that compose a shortest i-j path in D.
- $c(sp(i, j))$: Cost of a shortest i-j path in D.
- $z(i, s)$: Cost of solution of set s rooted in i.
- $j_s^*(i)$: Node at which set s splits, for an optimal $(i, l(s))$ structured directed Steiner tree.
- $N(s)$: Nodes that can be root node for subset s. Note that $N(s) = N$ for all $s \in S(l)$, with $s \subset K$. For $s = K$ we have $N(s) = r$.

Observe that, for all $k \in K$, $z(i, k) = c(sp(i, t_k))$. Since we only focus on laminar families whose tree representation is a full binary tree (see Proposition 2 of Chapter 2), then each set with at least 2 elements has exactly 2 children. We denote s_1 and s_2 to be the child sets of set $s \in S(l)$, $|s| \geq 2$. As an abuse of notation, we define $x(i, s)$ as follows $x(i, s) = sp(i, j) + x(j, s_1) + x(j, s_2)$, which means that the solution of the directed Steiner tree, rooted in i using sets in s, is the union of a shortest i-j path, and the solutions of its two child sets s_1 and s_2 rooted in j, where j is the splitting node of set s.

The recursion to compute $z(i,s)$ and $x(i,s)$ is the following,

$$z(i,s) = \min_{j \in N} \Big\{ c(sp(i,j)) + z(j,s_1) + z(j,s_2) \Big\} \qquad \forall i \in N(s), s \in S(l) : |s| \geq 2$$

$$j_s^*(i) \in \operatorname*{argmin}_{j \in N} \Big\{ c(sp(i,j)) + z(j,s_1) + z(j,s_2) \Big\} \qquad \forall i \in N(s), s \in S(l) : |s| \geq 2$$

$$x(i,s) = sp(i, j_s^*(i)) + x(j_s^*(i), s_1) + x(j_s^*(i), s_2) \qquad \forall i \in N(s), s \in S(l) : |s| \geq 2$$

Note that for $s = K$, we only need to compute $z(r,K)$ and $j_K^*(r)$. Let us assume that $S(l)$ is ordered in increasing order of the cardinality of its elements. Then, the algorithm to solve $\mathcal{Z}_l$ is the following,

Algorithm 2 Compute $x(i,s)$ and $z(i,s)$ for $s \in S(l), |s| \geq 2$

1: Set $x(i,k) = sp(i,t_k)$ for all $k \in K, i \in V$.
2: Set $z(i,k) = c(sp(i,t_k))$ for all $k \in K, i \in V$.
3: Set $z(i,s) = +\infty$ for all $s \in S(l), |s| \geq 2, i \in N$
4: **for** $s \in S(l), |s| \geq 2$ **do**
5: **for** $i \in N(s)$ **do**
6: **for** $j \in N$ **do**
7: **if** $c(sp(i,j)) + z(j,s_1) + z(j,s_2) < z(i,s)$ **then**
8: $x(i,s) \leftarrow sp(i,j) + x(j,s_1) + x(j,s_2)$
9: $z(i,s) \leftarrow c(sp(i,j)) + z(j,s_1) + z(j,s_2)$
return $x(r,K)$ and $z(r,K)$.

First, note that since the elements in $S(l)$ are ordered in increasing order of their cardinality, then for all $s \in S(l)$ with $|s| \geq 2$ we compute $x(i,s_1)$ and $x(i,s_2)$ before computing $x(i,s)$ and its cost $z(i,s)$, since $|s_1| < |s|$ and $|s_2| < |s|$. Now, the *for loop* in step 5 is looping over all possible root nodes for set s. The *for loop* in step 6, loops over all splitting nodes for set s. In step 7, we check whether the current best solution can be improved. If the solution can be improved, then the best solution is updated. Finally, the solution is given by $x(r,K)$ whose cost if $z(r,K)$.

Note that the number of sets in $S(l)$, with at least 2 elements, is $b-1$, since the tree representation of l is a full binary tree. Moreover, we have to visit at most n^2 pairs of nodes for each set in $S(l)$, with cardinality of at least 2, where $n = |V|$. Consequently, if the number of terminals is fixed, the complexity of the proposed algorithm is $\mathcal{O}(n^2)$.

One can think of this algorithm as a simplified version of the algorithm proposed in [30]. The main difference, is that the algorithm proposed [30], has to decide the way each set is split, while in this case, since we are in a particular laminar family, the split for every set is fixed.

3.2.3 Solution Improvements

For a given laminar family $l \in \mathcal{L}_b$, an optimal solution to $\mathcal{Z}_l$ may not correspond to a directed Steiner tree, but in the support of the solution we may have a directed Steiner tree, with a different structure, i.e., from a different laminar family in $\mathcal{L}_b$ (see Proposition 1 of Chapter 2).

Since the cost vector is positive, then for every set $s \in S(l)$, there are no cycles in any optimal solution of $\mathcal{Z}_l$. Consequently, the only way the solution is not a directed Steiner tree is when, at least, two different sets reach the same node. Following the notation of Chapter 2, if $x = (f, \hat{y}, \overline{y}, w)$ is an optimal solution for $\mathcal{Z}_l$, then there exists $i \in V$, such that for two sets $s_1, s_2 \in S(l)$, we have that $\sum_{a \in \delta^-(i)} f_a^{s_1} = 1$ and $\sum_{a \in \delta^-(i)} f_a^{s_2} = 1$. Whenever this happens, we have that the support of f vector does not correspond to a directed tree rooted in r.

By construction of the solution, there exists at least 1 path from r to t_k for all $k \in K$. Therefore, we can construct an r-arborescence using the arcs of the support of f, such that all the terminal nodes t_k are reached from r, and that all leaves of such arborescence correspond to terminal nodes. Let $A(l) = \{a \in A : \sum_{s \in S(l)} f_a^s \geq 1\}$ be the support of an optimal solution x to $\mathcal{Z}_l$, and let $T \subseteq A(l)$ be the set of arcs used by a minimum cost r-arborescence constructed in the subgraph defined by $A(l)$. If all leaves of T are terminal

nodes, then T is a directed Steiner tree. On the other hand, if at least 1 leaf is a Steiner node, then we can prune such leaves and still have an r-arborescence with a path connecting r to t_k for all $k \in K$. If the tree after the first round of pruning still has leaves that are Steiner nodes, then we can prune such a tree again. After a finite number of rounds of pruning, all leaves of the r-arborescence will correspond to terminal nodes, which will correspond to a directed Steiner tree.

Once we have a rooted Steiner tree, we can identify its structure. We just have to see the way the paths from the root r to every terminal t_k, share arcs. Since we are considering admissible laminar families, then the set of all commodities and all the singletons are always present. Consequently, by identifying the nodes where a split occurs, we can determine into which sets each set is partitioned, and therefore, identify the parent-child relationship between sets.

It may happen that the laminar family of such a tree may not have a full binary tree representation, or in other words, we may have that a set has more than 2 children. In this case, there are several laminar families in $\mathcal{L}_b$ that have the constructed solution as a feasible solution. When this happens, we randomly select one of those laminar families as the laminar family of the constructed solution, using Algorithm 3. Lets call the selected laminar family $\hat{l}$. Finally, since the constructed solution may not be an optimal solution for $\hat{l}$, we solve the sub-problem $\mathcal{Z}_{\hat{l}}$ to get a new solution.

We describe the algorithm to construct a random laminar family from a laminar family, whose tree representation is not a full-binary tree. As an abuse of notation, we refer to a laminar family l as the collection of sets contained in such family, denoted by $S(l)$.

Algorithm 3 Algorithm $randomLaminar(l)$

Require: Collection of sets $S(l)$.

1: Set $S' = \{s \in S(l) : s \text{ has at least 3 children}\}$
2: **while** $S' \neq \emptyset$ **do**
3: Select arbitrary $\hat{s} \in S'$
4: Let $\hat{S} = \{s \in S(l) : s \text{ is child set of } \hat{s}\}$
5: **while** $|\hat{S}| \geq 3$ **do**
6: Let s_1 and s_2 be randomly selected elements of $\hat{S}$
7: $s' \leftarrow s_1 \cup s_2$
8: $S(l) \leftarrow S(l) \cup s'$
return $S(l)$.

Algorithm 3 creates a laminar family $\hat{l}$ with full-binary tree representation from an admissible laminar family l, whose tree representation is not a full-binary tree. For every set $\hat{s}$ in $S(l)$ that has at least 3 children, we take two random children s_1 and s_2 (step 6), and we create a new set from the union of s_1 and s_2, which is added to $S(l)$ (steps 7 and 8). Note that this reduces the number of child sets of $\hat{s}$ by 1, since $s_1 \cup s_2$ is now a child of $\hat{s}$, while s_1 and s_2 are now children of $s_1 \cup s_2$.

3.3 Laminar family neighborhood characterization

Our final goal is to propose a local search algorithm to solve the directed Steiner tree problem using the LP-based approach proposed in Chapter 2. This local search algorithm starts from a given laminar family, and then moves to a promising neighbor laminar family. This process is performed several times. Each laminar family subproblem can be solved in polynomial time using the algorithm presented in Section 3.2.

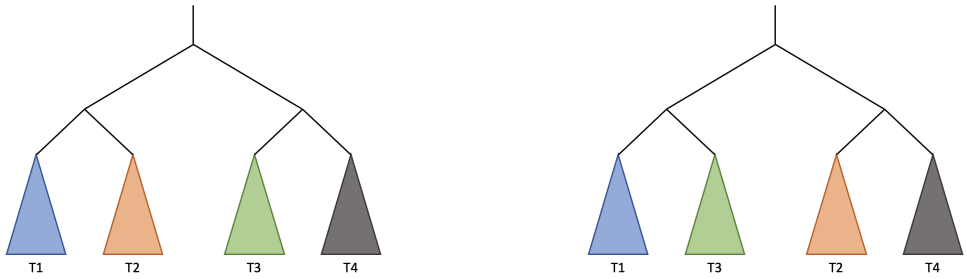
Consequently, we need to have a characterization of the neighborhood of each laminar family. Recall that, all the laminar families considered in the proposed approach have a unique corresponding full binary tree representation. Therefore, it suffices to define the

neighborhood of a laminar family, as the neighborhood of its corresponding tree representation.

In the literature, there are several ways proposed to characterize the neighborhood of a tree. The most commonly used are the Nearest Neighbor Interchange (NNI), the Subtree Pruning and Regrafting (SPR), and the Tree Bisection Reconnection (TBR) [12, 41].

For the rest of this section, we define T to be a full binary tree, i.e., each node in T has a degree 3 or 1.

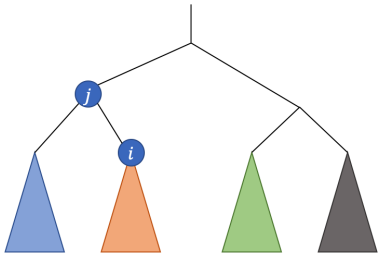
The NNI process consists in selecting two subtrees within tree T, and then swapping them as shown in Figure 3.2



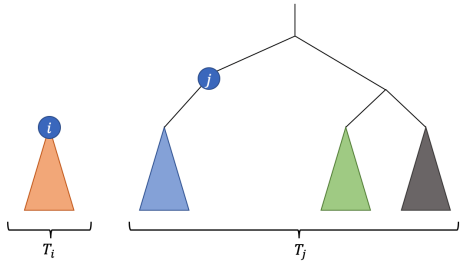
(a) Tree T. (b) Neighbor of tree T using NNI procedure.

Figure 3.2: A tree T and a neighbor under the NNI procedure. Subtrees $T2$ and $T3$ were swapped.

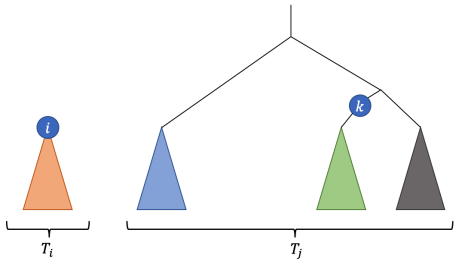
The SPR process corresponds to selecting an edge $\{i, j\}$ of tree T. The edge $\{i, j\}$ is removed from T, dividing T in two connected subtrees T_i and T_j, containing i and j respectively. For simplicity of the argument, suppose that T_i and T_j have more than 3 nodes. Note that nodes i and j will have degree 2, after removing edge $\{i, j\}$. We leave T_i as is, and we replace the two incident edges to j in T_j, by an edge connecting the two neighbors of j in T_j. Then, we select an edge of the updated T_j, and then we subdivide it creating a new node k. Finally, we create a new edge $\{i, k\}$ connecting node i of subtree T_i, and node k of subtree T_j.



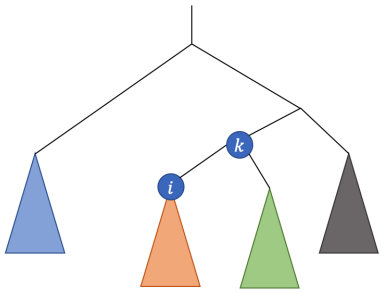
(a) Tree T.



(b) Subtrees T_i and T_j, after removing edge $\{i, j\}$.



(c) Original subtree T_i, and new subtree T_j.



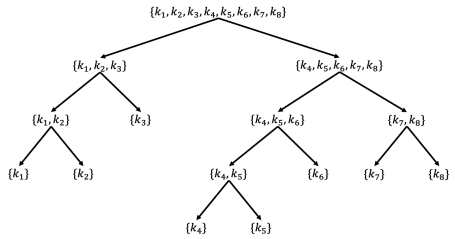
(d) Neighbor of tree T using SPR procedure.

Figure 3.3: A tree T and a neighbor under the SPR procedure.

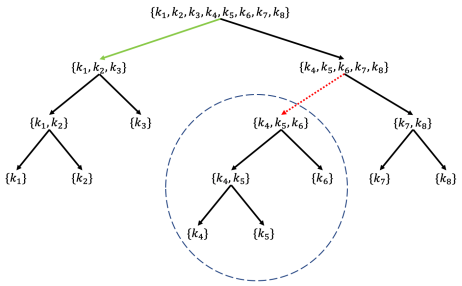
Finally, the TBR process is a generalization of SPR. We select an edge $\{i, j\}$ of tree T, and we remove such edge from T, leaving two connected subtrees T_i and T_j, containing i and j respectively. Again, for simplicity of the argument, suppose that T_i and T_j have a least 3 nodes. The main difference with SPR, is that in TBR, we replace the two edges incident to i in T_i by an edge connecting the two neighbors of node i in T_i, and we do the same for T_j. Then, we select an edge from T_i, and we subdivide it creating a new node l, and we do the same for T_j, but creating a node k. Finally, we create a new edge $\{l, k\}$ connecting T_i and T_j.

3.3.1 SPR neighborhood of laminar families

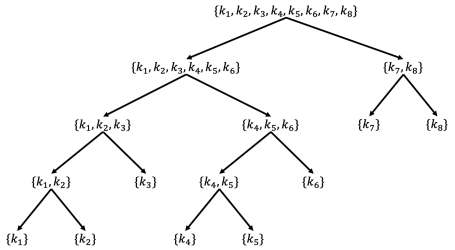
We decide to use the SPR process to construct the neighbors of a laminar family. In the studied literature, SPR was widely used over NNI and TBR. Furthermore, we consider the neighborhood size under SPR, which is $4(b-2)(b-3)$ [41], to be adequate for our case. Finally, we consider that SPR is reasonably easy to be adapted to our setting. Figure 3.4 shows an example of the SPR process applied to our setting.



(a) Original tree T.



(b) The subtree within dashed circle is pasted in green edge. The red dashed edge is removed.



(c) SPR neighbor of tree T.

Figure 3.4: SPR neighbor of a laminar family.

In Figure 3.4a, we can see the original tree representation of a given laminar family. In Figure 3.4b, we can see the subtree which is going to be removed and then regrafted, which is highlighted with the blue dashed circle. The red dashed edge is going to be removed, and the green edge is where the subtree is going to be regrafted. Figure 3.4c shows the SPR neighbor, note that one set was removed $\{k_4, k_5, k_6, k_7, k_8\}$, and one was added $\{k_1, k_2, k_3, k_4, k_5, k_6\}$. This has to be done since the neighbor is the tree representation of a laminar family.

Note that we only need to make a few extra computations to have the optimal solution to the laminar family shown in Figure 3.4c. Since we are using the algorithm presented in Section 3.2, we only need to compute $z(i, \{k_1, \ldots, k_6\})$ for all $i \in V$, and recompute $z(r, K)$, since the child sets of $K = \{k_1, \ldots, k_8\}$ are different in the new laminar family. Everything else can be reused from the computations done to get an optimal solution of laminar family shown in Figure 3.4a.

3.4 Simulated annealing

Simulated annealing is a metaheuristic, which is widely used in local search frameworks [57, 104, 1, 8]. The main drawback when using local search algorithms, is that we often get stuck in local optimums. In minimization problems, at each iteration, local search algorithms look at the neighborhood of the current solution, and then decide to move to a solution whose cost is lower than the current solution. In a local optimum, there are no better neighbors, and then the local search ends.

To address this problem, simulated annealing also allows movements to neighbors with higher cost with a certain probability, which depends on how much worse the new the solution is, and the iteration of the algorithm. The high-level idea of simulated annealing is the following. We set an initial temperature T_0, and a starting initial solution. Then, at each iteration j, we decrease the temperature of the system having $T_j = f(T_0, j)$, where f is a function depending on T_0 and j. Also, at each iteration we randomly choose a solution in the neighborhood of the current solution, and we compute the cost difference between the current solution and the neighbor, denoted by Δ_j. If $\Delta_j < 0$ then we move to the neighbor solution. If $\Delta_j \geq 0$ we move to the neighbor solution with probability $p(\Delta_j, T_j)$. Note that the probability is a function of the difference in cost of the solutions and the current temperature of the system.

There are three main decisions to make when using simulated annealing. How we set the initial temperature T_0, which function we use to reduce the temperature of the system, and what probability function we use to accept increasing cost neighbors. We decide to use the following probability function

$$p(\Delta_j, T_j) = \left[1 + exp\left(\frac{\Delta_j}{T_j}\right)\right]^{-1} \in \left(0, \frac{1}{2}\right)$$

Note that higher Δ_j leads to lower $p(\Delta_j, T_j)$, and smaller T_j leads to lower acceptance probability.

Since we want Δ_j and T_j to be in the same order of magnitude, we decide to use the cost of the first solution found as T_0. Finally, we use, by default, the following temperature cooling function

$$T_j = f(T_0, j) = T_0(0.95)^j$$

Algorithm 4 Simulated Annealing framework to solve directed Steiner tree problem

1: Set N_{iter}, $j = 1$.
2: $l_{current} \leftarrow initial\_laminar()$, $x_{current} \leftarrow solve\_DP(l_{current})$, $c_{current} \leftarrow c(x_{current})$, $T_0 \leftarrow c_{current}$
3: $l_{best} \leftarrow l_{current}$, $x_{best} \leftarrow x_{current}$, $c_{best} \leftarrow c_{current}$
4: **while** $j \leq N_{iter}$ **do**
5: Set $T_j = f(T_0, j)$
6: $l_{new} \leftarrow SPR(l_{current})$, $x_{new} \leftarrow solve\_DP(l_{new})$, $c_{new} \leftarrow c(x_{new})$
7: **if** $c_{new} < c_{current}$ **then**
8: $l_{current} \leftarrow l_{new}$, $x_{current} \leftarrow x_{new}$, $c_{current} \leftarrow c_{new}$
9: **if** $c_{new} < c_{best}$ **then**
10: $l_{best} \leftarrow l_{new}$, $x_{best} \leftarrow x_{new}$, $c_{best} \leftarrow c_{new}$
11: **else**
12: $\Delta_j = c_{new} - c_{current}$
13: Sample $u \sim U(0, 1)$
14: **if** $u \leq p(\Delta_j, T_j)$ **then**
15: $l_{current} \leftarrow l_{new}$, $x_{current} \leftarrow x_{new}$, $c_{current} \leftarrow c_{new}$
16: $j \leftarrow j + 1$
return x_{best}

The first 3 steps correspond to initialization of the algorithm. In the first step, we define the number of iterations N_{iter}, and set the iterations counter j to 1. In the second step, we

create an initial laminar family using Algorithm 5, then we compute the best solution to the given laminar family, denoted by $x_{current}$, and the cost of the solution. We also define the initial temperature value. In the third step, we save the current solution as the best solution found so far.

Then, for each iteration j we, first, compute the temperature given by function $f(T_0, j)$, and after that, we create a random SPR neighbor of the current laminar family, and we compute the best solution of the neighbor and its cost (step 6). We check whether the solution of the neighbor is better than the current solution, and if it is, we update the current solution. Moreover, if the neighbor solution is better than the best solution found so far, we update the best solution (steps 7 to 10). If the solution of the neighbor is worse than the current solution, we compute the cost difference between the solutions and we sample a uniform $(0, 1)$ random variable u. If the value of u is less or equal than the acceptance probability given by $p(\Delta_j, T_j)$, then we update the current solution (steps 11 to 15). Finally, we increase the iterations counter by 1.

The process to construct the initial structure to solve is the following. First, we create a complete graph G' whose nodes are the terminal nodes. For every edge $e' = (t', t'')$, its cost is the length of the shortest path between t' and t'' in the original graph G. Second, we construct a minimum spanning tree T' in G'. Finally, we run Algorithm 5 to construct an initial laminar family using T' as input.

Algorithm 5 Algorithm $initial\_laminar()$, which creates an initial laminar family

Require: Tree T', we assume T' is a set of edges ordered in increasing order of their lengths

Set $S(l) \leftarrow \{\{k_1\}, \{k_2\}, \ldots, \{k_b\}\}$

for $e' \in T'$ **do**

    Let $e' = (t', t'')$, and let k' and k'' be the commodities of terminals t' and t'', respectively.

    Let s' be largest set in $S(l)$ containing k'

    Let s'' be largest set in $S(l)$ containing k''

    $\hat{s} \leftarrow s' \cup s''$

    $S(l) \leftarrow S(l) \cup \hat{s}$

return $S(l)$.

Algorithm 5 constructs a laminar family l based on the single linkage clustering algorithm, which is a widely used aglomerative clustering method [112]. It starts with the collection $S(l)$ of all the singletons. Then, it goes over the set of all edges of T', which are assumed to be ordered in increasing order of their lengths. For each edge $e' = (t', t'')$, we take the commodities k' and k'' of terminals t' and t'', respectively. Then, we take sets s' and s'', which are the largest sets in $S(l)$ that contain k' and k'', respectively. Finally, we create a set $\hat{s}$ to be the union of s' and s'', and we add it to $S(l)$. After we visit all edges, we have that $S(l)$ is a collection of sets that forms an admissible laminar family.

3.4.1 Simulated annealing with solution improvement

As previously pointed out, sometimes an optimal solution to a laminar family sub-problem may not be a directed Steiner tree, which is why it may be beneficial to improve the solution obtained at every iteration of the simulated annealing algorithm. We use the solution improvement routine introduced in Section 3.2.3 in the simulated annealing framework, which is presented below.

Algorithm 6 Simulated Annealing for directed Steiner tree problem with solution improvement

Set N_{iter}, $j = 1$.
$l_{current} \leftarrow random\_laminar()$, $x_{current} \leftarrow solve\_DP(l_{current})$, $c_{current} \leftarrow c(x_{current})$, $T_0 \leftarrow c_{current}$
$l_{best} \leftarrow l_{current}$, $x_{best} \leftarrow x_{current}$, $c_{best} \leftarrow c_{current}$
while $j \leq N_{iter}$ **do**
    Set $T_j = f(T_0, j)$
    $l_{new} \leftarrow SPR(l_{current})$, $x_{new} \leftarrow solve\_DP(l_{new})$, $c_{new} \leftarrow c(x_{new})$
    if $A(l_{new})$ is not an r-arborescence **then**
        $T \leftarrow MST(A(l_{new}))$
        $l_{improved} \leftarrow LaminarFamily(T)$
        if $l_{improved}$ is not full-binary tree **then**
            $l_{improved} \leftarrow SampleFullBinaryTree(l_{improved})$
        $l_{new} \leftarrow l_{improved}$, $x_{new} \leftarrow solve\_DP(l_{new})$, $c_{new} \leftarrow c(x_{new})$
    if $c_{new} < c_{current}$ **then**
        $l_{current} \leftarrow l_{new}$, $x_{current} \leftarrow x_{new}$, $c_{current} \leftarrow c_{new}$
        if $c_{new} < c_{best}$ **then**
            $l_{best} \leftarrow l_{new}$, $x_{best} \leftarrow x_{new}$, $c_{best} \leftarrow c_{new}$
    else
        $\Delta_j = c_{new} - c_{current}$
        Sample $u \sim U(0, 1)$
        if $u \leq p(\Delta_j, T_j)$ **then**
            $l_{current} \leftarrow l_{new}$, $x_{current} \leftarrow x_{new}$, $c_{current} \leftarrow c_{new}$
    $j \leftarrow j + 1$
return x_{best}

Algorithm 6 shows the pseudocode of simulated annealing with the solution improve-

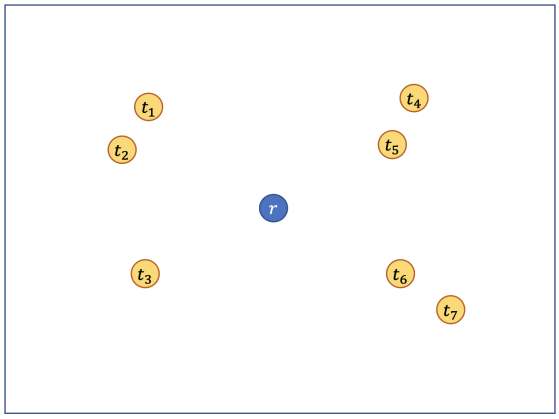
ment. The solution improvement, or tester, corresponds to lines 7 to 12. First, we check whether the solution of l_{new} is an r-arborescence or not, in case it is not, then it can be improved. In step 8, we define T to be the minimum cost r-arborescence whose leaves correspond to terminal nodes, as described above. In line 9, we define $l_{improved}$ to be the laminar family extracted from T. In line 10, we check whether $l_{improved}$ has a full binary tree representation or not. In case it is not full-binary, then we sample a full binary tree that contains $l_{improved}$. Finally, we update l_{new}, x_{new} and c_{new}.

3.4.2 Rectilinear graphs

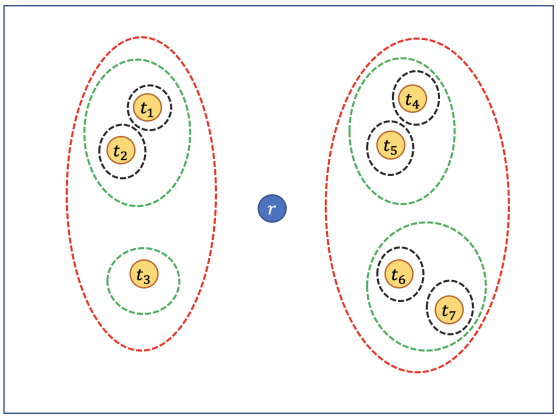
The rectilinear graphs are a special class of graphs, where the nodes are placed in the $\mathbb{R}^2$ plane and the distance between nodes is given by the $\|\cdot\|_1$ distance. We can take advantage of this property since we can divide the plane into regions, each one containing a set of terminals, which will be used to create an initial laminar family. For instance, in Figure 3.5a, we can see that terminals t_1 and t_7 may not share arcs in an optimal solution, but it is very likely that t_1 shares arcs with t_2, and t_7 with t_6, since they are in similar regions of the plane.

The idea is to first, divide the set of terminal nodes in two sets, which will be interpreted as the first bipartition for the laminar family we want to construct. Then, each set of commodities is divided into another two sets, and so on, until all sets are singletons. In particular, the first partition corresponds to the partition given by clusterizing all the terminals into two clusters. Then each cluster is clusterized again into another two clusters, and so on, until each cluster has size 1. To illustrate this idea see Figure 3.5. In Figure 3.5a, we can see a rectilinear graph, where we only show the root node and the terminal nodes. Figure 3.5b shows a potential partition based on the euclidean distance between terminals. In red dashed ovals, we find the first bipartition, inside each of the red zones, we find the second bipartition which is delimited by the green dashed oval, finally, inside each green zone with at least 2 terminals, we find a new partition denoted by the black dashed circles.

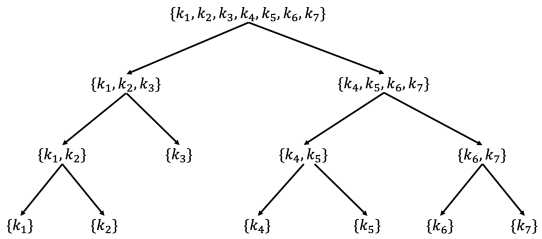
Figure 3.5c shows the tree representation of the laminar family associated with the partition shown in Figure 3.5b.



(a) Rectilinear graph, only showing root and terminal nodes



(b) Example of a clusterization of terminals in rectilinear graph



(c) Tree representation of laminar family of clusterization

Figure 3.5: Example of terminals clusterization

There are many algorithms to cluster elements in the plane [54]. We decided to use k-means algorithm [73] which is widely used in practice, because it is easy to implement, it runs very quickly, and it performs well in these type of clustering problems. Since k-means output depends on the initial centroids, then the quality of our constructed laminar family, when using k-means as clustering algorithm, will also depend on the initial centroids of each clusterization. This is why, we run the clustering-based algorithm several times to create laminar families, and then we select the laminar family with the best solution among

all the candidates.

Algorithm 7 Algorithm $Part(s)$, which creates a bipartition of input set s, and all the created subsets

Require: Set s.

1: Define $\hat{S} = \{s\}$.
2: **if** $|s| = 1$ **then return** $\hat{S}$
3: **else if** $|s| = 2$ **then**
4: $\quad \hat{S} \leftarrow \hat{S} \cup \{\{e_1\}, \{e_2\}\}$, where $s = \{e_1, e_2\}$
5: **else**
6: $\quad (s_1, s_2) = kMeans(s, 2)$
7: $\quad \hat{S}_1 = Part(s_1)$, $\hat{S}_2 = Part(s_2)$
8: $\quad \hat{S} \leftarrow \hat{S} \cup \hat{S}_1 \cup \hat{S}_2$
return $\hat{S}$

Algorithm 7, named $Part(s)$, determines the way a new laminar family is created based on the clustering process previously described. This algorithm takes as input a given set s. If the set has only one element, then a collection of sets $\hat{S}$, containing the singleton set s, is returned as shown in line 4. If set s has exactly 2 elements, then the algorithm returns a collection of sets $\hat{S}$, containing set s and the two singletons, as shown in line 6. If set s has 3 or more elements, then s is split into two subsets s_1 and s_2, which is the result of running k-means algorithm for $k = 2$, i.e., two clusters. Then, the algorithm recurses getting two collections of sets $\hat{S}_1$ and $\hat{S}_2$, which is the result of applying algorithm $Part(\cdot)$ to s_1 and s_2 respectively. The returned collection of sets $\hat{S}$, contains all sets in $\hat{S}_1$, all sets in $\hat{S}_2$, and the set s. Consequently, to get the desired laminar family, we have to run algorithm $Part(s)$, for $s = K$.

When the instance we want to solve is undirected, then we need to choose the root node among the terminal nodes. For all $i \in R$, let $L(i)$ and $R(i)$ be the number of terminal nodes to the left of i, and to the right of i, respectively. And, let $U(i)$ and $B(i)$ be the number of terminals that are above i, and below i in the plane, respectively. Finally, let

$\Delta_x(i) = |R(i) - L(i)|$ and $\Delta_y(i) = |U(i) - B(i)|$, then we choose r as follows.

$$r = \underset{i \in R}{\operatorname{argmin}}\{\Delta_x(i) + \Delta_y(i)\}$$

We are basically choosing r to be the most centrical terminal node. We pick r in this fashion, since we can have a better guess of the structure of an optimal directed Steiner tree. For instance, in the example shown in Figure 3.5a, it is very likely that the paths from r of the terminals that are above and to the left of r, i.e., terminals t_1 and t_2, are not going to share many arcs with the paths from r to terminals that are below and to the right of r, i.e., terminals t_6 and t_7.

For rectilinear graphs, we use algorithm 7 to create the initial laminar family in the simulated annealing framework. As described in Algorithms 4 and 6, the initial laminar family is created at random. In rectilinear graphs, we run Algorithm 7 several times (the number of clusterizations is a parameter of the algorithm), and then we select a laminar family from the set of laminar families with with the lowest optimal cost. The rest of the algorithm is the same as algorithm 4, or algorithm 6 if we want to use the solution improvement. Moreover, if the instance we want to solve is undirected, we choose r as the most centrical terminal, as previously described.

3.4.3 Worst case analysis

Let OPT be the value of an optimal solution for the directed Steiner tree problem instance, let OPT_{SA} be the value of the solution returned by the simulated annealing framework, and for $l \in \mathcal{L}_b$, let $OPT(l)$ be the value of an optimal solution to $\mathcal{Z}_l$.

Lemma 2. *For any $l \in \mathcal{L}_b$ we have $OPT(l) \leq |R| \times OPT$*

Proof. Let l be an arbitrary laminar family of $\mathcal{L}_b$. Note that we can always construct the following feasible solution. For all $s \in S(l)$ with $|s| \geq 2$, we fix $f_a^s = 0$ for all $a \in A$, and for all $k \in K$, which correspond to $s \in S(l)$ with $|s| = 1$, we take a shortest path from r

to t_k. It is known that such solution is at most $|R|$ times the value of an optimal solution of the problem [100]. Consequently, any optimal solution to $\mathcal{Z}_l$ is at most $|R|$ times the value of an optimal solution to the problem. □

Proposition 3. *For every instance of the directed Steiner tree problem, we have* $OPT_{SA} \leq |R| \times OPT$

Proof. Since at every iteration we solve to optimality $\mathcal{Z}_l$ for some $l \in \mathcal{L}_b$, then using Lemma 2, we have that at every iteration, the best solution found cannot be more than $|R|$ times the value of an optimal solution to the problem, and the statement holds. □

Let $T(l)$ be the tree representation of laminar family $l \in \mathcal{L}_b$, and let $l^* \in \mathcal{L}_b$ be the laminar family of an optimal solution to the problem . We define $d_{SPR}(T(l_1), T(l_2))$ as the minimum number of SPR moves to transition from tree $T(l_1)$ to tree $T(l_2)$. It has been proven that computing $d_{SPR}(T(l_1), T(l_2))$ for any pair of l_1, l_2 is NP-Hard [11]. Nevertheless, it was proven in [97], that for any two $l_1, l_2 \in \mathcal{L}_b$, we have that $d_{SPR}(T(l_1), T(l_2)) \leq |R| - 2$. Consequently, if we use at least $|R| - 2$ iterations in the simulated annealing framework, no matter the initial laminar family chosen, the probability of solving $\mathcal{Z}_{l^*}$ in a given iteration, is strictly positive.

3.5 Computation experiments

In this section we present our computational results. We compare our proposed simulated annealing framework with all the algorithms studied in [107].

In [107], 6 algorithms are compared. The first algorithm, denoted by ShP_1, takes a shortest path from r to each terminal, and then returns the union of such shortest paths as a solution. The second algorithm, denoted by ShP_2, takes a shortest path from r to its closest terminal, sets the cost of all used arcs to 0, and then proceeds in the same fashion with the rest of the terminals, until all terminals are reached. The third algorithm, denoted by DuAs, corresponds to using the dual ascent algorithm presented in [109]. If such solution

is fractional, then it takes a minimum spanning tree within the support of the fractional solution. If some of the leave nodes are non-terminal nodes, then the tree is pruned until all leaves are terminal nodes. The fourth algorithm, denoted by Roos, corresponds to the algorithm presented in [16], which has an approximation ratio $\mathcal{O}\left(|R|^{\frac{1}{t}}\right)$ using $t = 2$, and is implemented using Roos modified algorithm which is described in [77]. The fifth algorithm, denoted by FLAC, is one of the algorithms introduced by the authors in [107]. This algorithm takes each arc as a pipe with capacity, in liters, equal to its cost. Then the algorithm tries to send water to the terminals, at a rate of 1 liter of water per second. Initially, only the arcs incoming to terminals are going to be considered. When an arc is up to its capacity, it is said to be saturated. Once an arc is saturated, then we start looking at the arcs incoming to the tail node of the saturated arcs too. This process is done until we reach the root node. Then, within the support of the saturated arcs, we have a directed Steiner tree. The sixth algorithm, denoted by FLAC$^{\triangleright}$, is the FLAC algorithm applied to the shortest path instance of the problem, i.e., to a complete directed graph where the cost from u to v is given by the shortest path, in the original graph, between u and v. Finally, we define the Best Benchmark (BB) to be the algorithm that, for each instance, takes the best result among the previous 6 algorithms. The solutions provided by our algorithms will be compared with the solutions provided by BB.

On the other hand, we present 3 algorithms, SA, SA-Test and SA-Rect algorithm. The SA algorithm corresponds to Algorithm 4, SA-Test corresponds to Algorithm 6, and SA-Rect corresponds to Algorithm 6 but using Algorithm 7 to construct the initial laminar family. Since all of the proposed algorithms have a random component, we run 10 replications of each one, and the result of each algorithm corresponds to the solution with the lowest cost among the 10 replications. Furthermore, we run each algorithm with 1,000 and with 5,000 iterations, since the number of iterations is a parameter of the proposed algorithms. In the SA-Rect case, we run 50 replications of Algorithm 7, and we select the solution with lowest cost as the initial laminar family of the simulated annealing algorithm.

We run experiments using the same instances studied in [107], which correspond to directed graphs constructed based on undirected instances from the SteinLib library [59]. We only consider instances with at most 160 terminal nodes, and less than 3,500 nodes, since the proposed approach requires to compute the shortest path between every pair of nodes. In total, we studied just over 800 instances, whose details can be found in Table B.1 of Appendix B.

The proposed algorithms were implemented in Java 8. All the experiments were run in an AWS *c5d.2xlarge* machine, with am *Intel Xeon Platinum 8000-series* processor (3.0 GHz) of 8 cores and 16 GB of memory.

3.5.1 Non-rectilinear graphs

In this section, we compare the solution quality of the regular simulated annealing algorithm (see Algorithm 4), the simulated annealing with solution improvement (see Algorithm 6), and the best benchmark algorithm. We only consider instances which are not rectilinear graphs, since there is a specific algorithm for such cases, which are analyzed in section 3.5.2. We use the performance profile approach to compare the solution quality delivered by the studied algorithms [29]. Figure 3.6 shows the cumulative distribution of instances versus the error of the obtained solution, this graph should be read as follows. If, for instance, we have a point $(1.5, 0.85)$ it means that 85% of the instances solved have a gap smaller or equal to 1.5%.

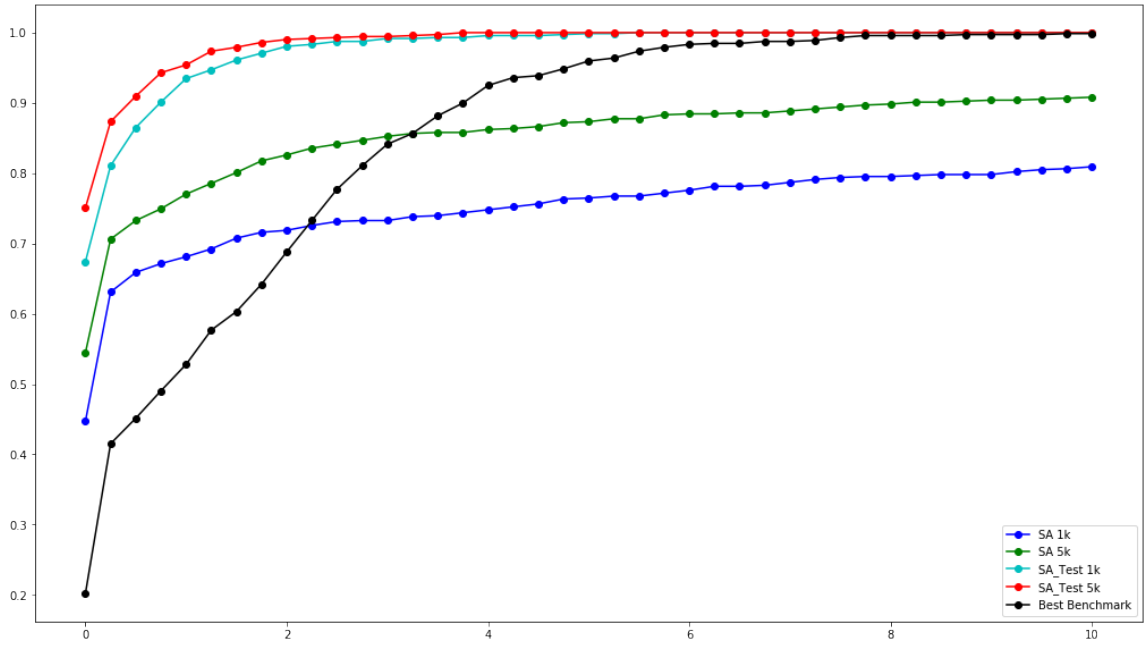


Figure 3.6: Performance profile for SA and SA-Test, for 1,000 and 5,000 iterations

It is clear from Figure 3.6 that SA-Test outperforms SA, and BB. First of all, we expected that, for the same algorithm, using more iterations will lead to better results. This happens with both, the SA and the SA-Test algorithm. The difference is more pronounced in the SA case; while in the SA-Test, the difference still exists, specially for smaller gaps, but after 2%, the two curves are almost identical. Also, note that SA-Test algorithm with just 1,000 iterations, outperforms by far the BB algorithm, and the SA algorithm with 5,000 iterations. Another interesting observation is the difference in the proportion of instances solved to optimality. The BB algorithm solves around 20% of the instances to optimality, being the worst of all the algorithms studied in this topic. Furthermore, while the SA algorithm cannot solve more than 55% of the instances to optimality, we have that SA-Test can solve over 67% of the instances to optimality with 1,000 iterations, and 75% of the instances with 5,000.

Table 3.1: Proportion of instances where SA-Test is strictly worse, equal, or strictly better than SA with 5,000 iterations

Algorithm	Worse than SA 5k	Equal to SA 5k	Better than SA 5k
SA-Test 1k	4.7%	55.4%	39.8%
SA-Test 5k	1.8%	55.8%	42.3%

Table 3.1 shows the proportion of instances where SA-Test algorithm does strictly worse, equal, or strictly better than SA with 5,000 iterations. We can see that even when we use 1,000 iterations, SA-Test only delivers worse results than SA in less than 5% of the cases studied, this number reduces to 1.8% when we use 5,000 iterations in SA-Test.

Table 3.2: Proportion of instances where SA-Test is strictly worse, equal, or strictly better than best benchmark

Algorithm	Worse than BB	Equal to BB	Better than BB
SA-Test 1k	3.6%	21.0%	75.3%
SA-Test 5k	2.2%	20.6%	77.2%

Table 3.2 show the proportion of instances where SA-Test algorithm does strictly worse, equal, or strictly better than BB. We can see that when we use 1,000 iterations, SA-Test only deliver worse results than BB in 3.6% of the cases studied, this number reduces to 2.2% when we use 5,000 iterations in SA-Test.

We conclude, that SA-Test outperforms, in solution quality, the BB and SA algorithms. Although SA-Test with 5,000 iterations presents a better performance than SA-Test with 1,000 iterations, the results are not considerably better.

3.5.2 Rectilinear graphs

In this section we consider rectilinear graphs. We compare the solution quality of the regular simulated annealing algorithm (see Algorithm 4), the simulated annealing for rectilinear graphs (see Algorithms 6 and 7), and the best benchmark algorithm. We compare the three algorithms in the same way we compared SA, SA-Test, and BB in section 3.5.1.

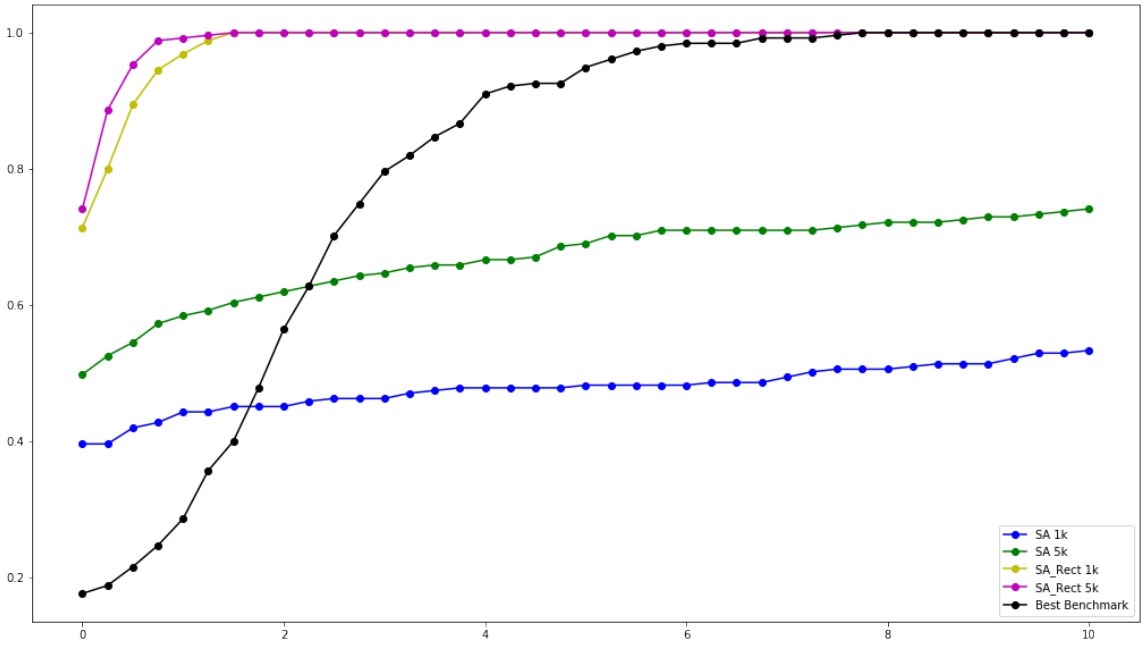


Figure 3.7: Performance profile for SA and SA-Rect, for 1,000 and 5,000 iterations

It is clear from Figure 3.7 that SA-Rect outperforms SA, and BB. The results are even more pronounced than the non-rectilinear graphs, since SA-Rect always delivers solutions with at most 1.75% gap. With respect to the SA and SA-Rect algorithms, again we see that by solving the problem with more iterations, the quality of the solutions improve. And again, the difference is more pronounced in the SA case. Indeed, for the SA-Rect the performance profiles are very similar, having a slightly better performance with 5,000 iterations.

In this case, we also have that SA-Rect algorithm with just 1,000 iterations, outperforms by far the BB algorithm, and the SA algorithm with 5,000 iterations. BB solves less than

18% of the instances to optimality, while SA solves almost 40% of instances to optimality with 1,000 iterations, and almost 50% with 5,000 iterations. In contrast, SA-Rect solves over 71% of instances to optimality with 1,000 iterations, and almost 75% with 5,000 iterations.

Table 3.3: Proportion of instances where SA-Rect is strictly worse, equal, or strictly better than SA with 5,000 iterations

Algorithm	Worse than SA 5k	Equal to SA 5k	Better than SA 5k
SA-Rect 1k	1.2%	49.8%	49.0%
SA-Rect 5k	1.2%	50.2%	48.6%

Table 3.3 shows the proportion of instances where SA-Rect algorithm does strictly worse, equal, or strictly better than SA with 5,000 iterations. We can see that when the number of iterations is 1,000, SA-Rect only delivers worse results than SA in 1.2% of the cases studied, the same amount when we use 5,000 iterations in SA-Rect.

Table 3.4: Proportion of instances where SA-Rect is strictly worse, equal, or strictly better than best benchmark

Algorithm	Worse than BB	Equal to BB	Better than BB
SA-Rect 1k	0.4%	17.3%	82.3%
SA-Rect 5k	0.0%	17.6%	82.4%

Table 3.4 shows the proportion of instances where SA-Rect algorithm does strictly worse, equal, or strictly better than BB. We can see that when we use 1,000 iterations, SA-Rect only delivers worse results than BB in 0.4% of the cases studied, which in this case corresponds to only one instance. When we use 5,000 iterations, SA-Rect performance is at least as well as BB in all the studied instances.

We conclude, that SA-Rect outperforms, in solution quality, the BB and SA algorithms.

Although the conclusions in the rectilinear case are similar to the non-rectilinear case, the solution quality obtained by the improved version of SA in rectilinear graphs is better than the non-rectilinear case. All the instances in the rectilinear case present a gap smaller or equal to 1.75%, while in the non-rectilinear case it is 3.5% when we run SA-Test with 5,000 iterations, and 5% with 1,000 iterations. Moreover, the proportion of instances where SA-Rect performs worse than, either SA or BB, is lower than SA-Test.

3.5.3 Execution times

At each iteration of simulated annealing, we compute the new solution based on the solution of the previous iteration. Therefore, simulated annealing, by nature, is a sequential algorithm. There are some researchers that study a parallel version of the algorithm [49, 87, 23], but we just focused on the original version of the algorithm. Consequently, the running time of the algorithm will depend on the number of iterations.

Figure 3.8 shows the histogram of the average execution time per iteration for simulated annealing with, and without the solution improvement routine.

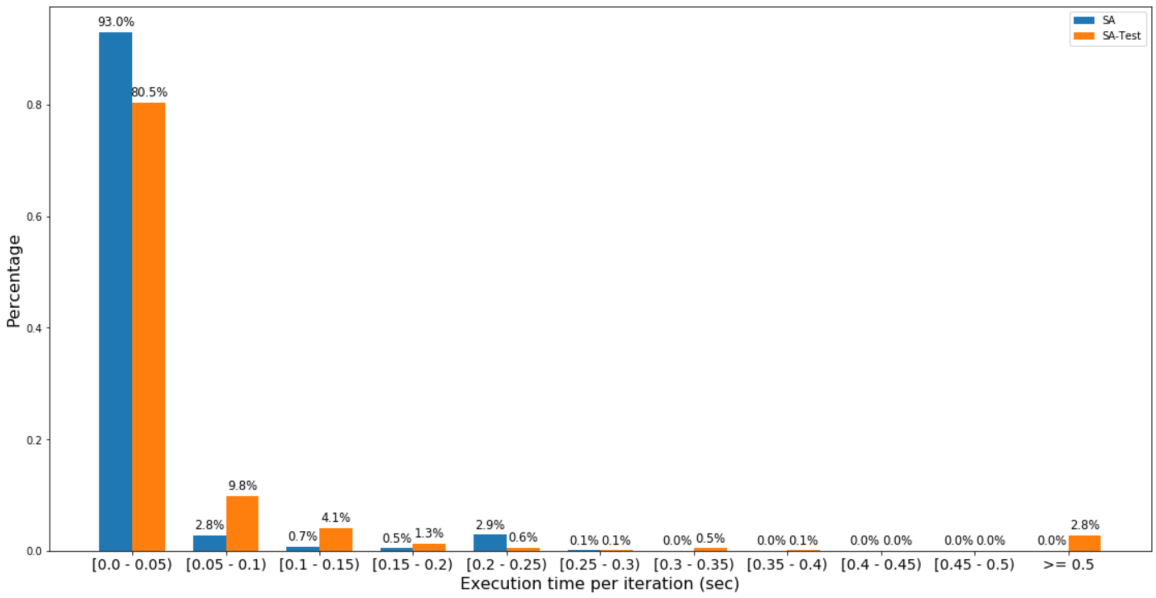


Figure 3.8: Execution time histogram for SA with and without solution improvement

As expected, the execution times when not using the solution improvement routine are lower, since there are fewer steps to complete at each iteration, but the distribution of the execution time when using solution improvement is not much worse in the studied instances. In both cases, the majority of the instances have an average execution time below 50 milliseconds.

The execution time per iteration, in both cases of the simulated annealing framework, are in the same order of magnitude as the execution times of all the studied algorithms in [107]. The main difference is that the total execution time of the algorithm is larger given the number of iterations to perform. In any case, the vast majority of the instances solve within seconds, or a few minutes, which makes this approach appealing to use given the better results obtained in terms of solution quality. Moreover, we can always use the solution given by the best benchmark algorithm, which takes a few milliseconds to solve, as the initial laminar family to consider, and start the simulated annealing framework from there.

3.6 Conclusions and future work

We develop a simulated annealing framework to solve the directed Steiner tree problem based the approach proposed in Chapter 2. We propose an efficient algorithm to solve the sub-problem of each tree structure, we provide a solution improvement algorithm, and we present a complete characterization of each tree structure. We compared the proposed framework against the algorithms studied in [107], and we conclude that our approach outperforms, in solution quality, the mentioned algorithms.

Future research might be directed in using the insights obtained in this chapter to other solution techniques for the problem. For instance, we can use the algorithm proposed to solve each laminar family sub-problem to find better upper bound in a Branch and Bound setting. In particular, we wonder if we can have an important reduction in execution times when we use the Branch and Ascent approach proposed in [3]. At any node of the search

tree, we can get candidate laminar families from the support of the fractional solution, and then solve the problem for that set of tree structures to compute better primal bounds.

CHAPTER 4
LOT TARGETING AND LOT DISPATCHING DECISION POLICIES FOR SEMICONDUCTOR MANUFACTURING: OPTIMIZATION UNDER UNCERTAINTY WITH SIMULATION VALIDATION

The work presented in this chapter has already been published [96].

4.1 Introduction

In this chapter, we develop a model and algorithms for optimizing *Lot Targeting* and *Lot Dispatching* decisions under the uncertainties of production. We use a discrete-event simulation model to evaluate the effects of these policies on AMHS productivity by comparing system throughput under our proposed rules with throughput under commonly used rules.

4.1.1 Our contributions

Our main contributions in this topic are summarized below

1. We propose a new fluid-model based lot dispatching policy that iteratively optimizes lot selection based on current work-in-progress (WIP) distribution of the entire system.

2. Furthermore, we propose to split the decision policies into two phases in order to include travel time information into the dispatching and targeting decisions.

3. We provide simulation results for a prototype facility that show that our proposed policies outperform commonly used dispatching rules in throughput, machine utilization and machine target accuracy.

The remainder of this chapter is organized as follows. Section 4.2 describes the problem we consider in this chapter. Section 4.3 briefly describes fluid models and presents our

proposed model and policies. Section 4.4 describes our simulation structure and results. Finally, section 4.5 concludes and offers suggestions for future research.

4.2 Problem Description

An important metric used to measure performance in semiconductor manufacturing is throughput. The utilization level of bottleneck machines drives this throughput level. In this chapter, we address two factors that affect machine utilization and cause unnecessary idle time: long lot travel times, and myopic lot selection policies.

After a *Lot Targeting* decision is made, the lot is transferred to the assigned machine if any of the target machine's input ports are currently available. We call this *machine-to-machine direct transfer*. If the assigned machine has no available input ports, the lot is sent to an intermediate storage buffer (primarily a Side Track Buffer (STB) or stocker) to make room for the next lot to be processed at the departure machine. The lot will then be called from storage by the *Lot Dispatcher* when a machine of the appropriate type becomes available. The machine that pulls a lot from a storage location, via *Lot Dispatching*, may not be the same machine selected for the lot by *Lot Targeting*. We call this *storage-through transfer*. Due to the complexities of wafer fab production and the material handling systems, there are sometimes unintentional long transfers that cause machines to be unnecessarily idle.

First, existing policies only consider the current state of the system without any sort of look-ahead at upcoming events. Even without the use of storage location, *long-distance direct transfers* may occur. In Figure 4.1, the lot leaving Machine 1 is sent to an input port of Machine 4 and, subsequently, the lot from Machine 2 is sent to an input port of Machine 3. If, instead, we knew that the lot at Machine 2 will be finished processing in 30 seconds, we may wait for Machine 3 to be available for the lot from Machine 1 and then send the lot from Machine 2 to Machine 4, reducing the load on the AMHS.

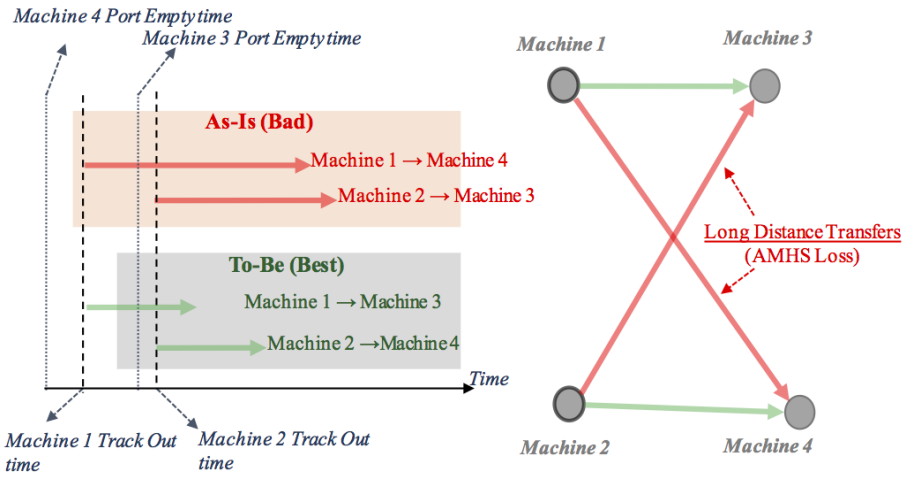


Figure 4.1: Long-distance direct transfer

Second, *Lot Targeting* and *Lot Dispatching* work independently. Though *Lot Targeting* selects a target machine before sending a lot to storage, *Lot Dispatching* does not consider the target machine when selecting a lot to pull from storage. This may cause what we refer as *long-distance storage-through transfer*. For instance, in Figure 4.2 the lot dispatched to Storage 2 for Machine 2 is transferred to Machine 3 because its port becomes idle earlier than any of Machine 2's ports and the total weight of the lot is the highest at the time Machine 3 triggers *Lot Dispatcher*.

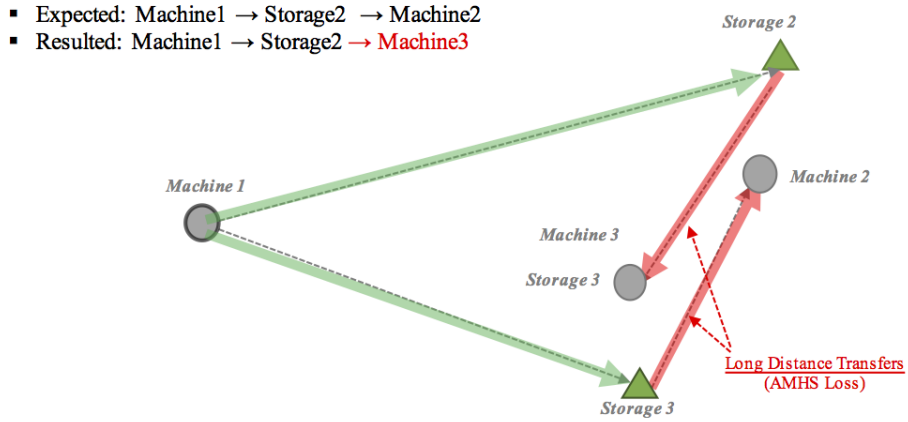


Figure 4.2: Long-distance storage-through transfer

Large-scale manufacturing facilities use several thousands of STBs to hold WIP be-

tween processing steps. Each STB has a capacity of just one lot and the STBs are located near or above machines. Stockers, on the other hand, have higher capacity and are located only near the center loop of the fabrication facility. With so many storage locations available, assigning lots to storage locations between process steps can be difficult; bad decisions can cause deterioration in transfer time and increase vehicle workload, thus leading to machine idle time.

Finally, we want to address the issue that commonly used dispatching policies are myopic, i.e., they only consider local information rather than the whole system's information. In consequence, the decisions made using these policies lead to good local solutions but not necessary good global decisions. For this reason, one of our goals is to develop decision policies that consider all the system's information in the decision making, which we expect to outperform commonly used decision policies in semiconductor manufacturing.

4.3 Proposed Model and Policies

As was mentioned in Section 4.2, our main goal is to maximize the system throughput by developing better lot selection policies than the commonly used ones, and by avoiding long-time wafer transfers. We propose lot selection policies based on the solution obtained by optimizing the underlying fluid model of the system. The second issue is addressed by splitting the *Lot Targeting* and *Lot Dispatching* decision into two phases, which allows us to include the travel times in the decision making.

4.3.1 Original Fluid Model

We base our solution on the fluid model idea proposed by [17] which was first applied to a semiconductor manufacturing problem by [22]. This approach views the problem as a resource allocation problem rather than a detailed job sequencing problem.

Fluid models correlate each step of the semiconductor manufacturing process to a fluid in the model. Each work station then processes only a particular subset of fluids. The

model decides the amount of time each station should devote to each fluid. For instance, if a given station can process steps 4, 10, and 22 of a given product, then the solution of the fluid model is the proportion of time that the station devotes to process each step, e.g., 30%, 25%, and 35% of the time, respectively. Since the sum is 90%, this station should work 90% of the time. We interpret the solution as follows. Using a 10-hour interval, the station works 3 hours on step 4, 2.5 hours on step 10, and 3.5 hours on step 22. Refer to Section 4.3.2 for an explanation of the order in which each step is processed.

The fluid model takes as input the state of the system comprising i) the current queue at each station, ii) the external rate at which a job is released into the system, and iii) the processing rate at which each station is working. The output given by the model is the proportion of time that each station devotes to a given fluid, or processing step, in the long-run. The solution is usually feasible for a limited time period because under the optimal solution given by the fluid model, some fluid levels increase over time and others decrease. The solution is feasible until the storage capacities are exceeded or the amount of a fluid becomes negative, at which point the model is re-optimized using the current state of the system as input.

4.3.2 Proposed Fluid Model

As mentioned in the introduction, most of the existing research focuses on the *Lot Dispatching* policies, but in practice decisions have to be made for the *Lot Targeting* as well. For example, in [22], the solution provided by the fluid model is the proportion of time that each station should devote to each fluid, which can be used to generate *Lot Dispatching* policies. Unfortunately, the solution is not detailed enough to develop *Lot Targeting* policies. Our model can be used in both *Lot Dispatching* and *Lot Targeting*, allowing alignment between the two decisions. This section details the mathematical formulation of the proposed fluid model.

Sets, Parameters, and Notations

M : Set of machines or nodes.

F : Set of distinct fluids, or processing steps.

$F(s)$: Subset of fluids that machine s can process.

$M(k)$: Subset of machines that can process fluid k.

P : Flow-transfer matrix.

- $P \in \mathbb{R}^{|F|\times|F|}$
- $p_{i,j}$: Proportion of fluid i that is transformed into fluid j.
- $p_{i,j} \geq 0 \quad \forall i, j \in F, \quad \sum_{j\in F} p_{i,j} \leq 1 \ \ \forall i \in C.$
- $1 - \sum_{j\in F} p_{i,j}$: Proportion of fluid i that exits the network after being processed in $M(i)$.

λ_k^s : Exogenous external rate at which fluid k arrives at machines s.

$\mu_k^{s,l}$: Constant rate at which fluid k is processed in machine s and then sent to machine l.

$T_k^{s,l}(t)$: Cumulative time up to time t that machine s uses to process fluid k and sends to machine l.

- $\mu_k^{s,l} T_k^{s,l}(t)$: Total amount of fluid k process by machine s and sent to machine l up to time t.

$Q_k^s(t)$: Total amount of fluid k in machine s at time t.

$$\alpha_k^{s,l} = \begin{cases} 1 & \text{if machine } s \text{ can process fluid } k \text{ and then send it to machine } l \\ 0 & \text{otherwise} \end{cases}$$

Model and Control Problem

The fluid model satisfies the following equation:

$$Q_k^s(t) = Q_k^s(0) + \lambda_k^s t + \sum_{j\in F}\sum_{l\in M} p_{j,k}\mu_j^{l,s}T_j^{l,s}(t) - \sum_{l\in M}\mu_k^{s,l}T_k^{s,l}(t) \quad \forall k \in F, s \in M \quad (4.1)$$

which states that the amount of fluid k in machine s at time t is equal to the initial amount of fluid k in machine s, plus the external amount of fluid k arriving at machine s, plus the sum of all the fluids that becomes fluid k and that are sent to machine s, minus all the fluid k processed at machine s up to time t.

Using the same reasoning of [17] and [22], we assume that the function $T_k^{s,l}(t)$ is a piecewise linear function, therefore $T_k^{s,l}(t) = x_k^{s,l}t$ where $x_k^{s,l}$ is the proportion of time that machine s devotes to process fluid k and that is sent to machine l. Under this assumption we have that $Q_k^s(t)$ is an affine function on t, and therefore we have that the variation on the amount of fluid k in machine s is constant over time.

$$Q_k^s(t) = Q_k^s(0) + \lambda_k^s t + \sum_{j\in F}\sum_{l\in M} p_{j,k}\mu_j^{l,s}x_j^{l,s}t - \sum_{l\in M}\mu_k^{s,l}x_k^{s,l}t \quad \forall k \in F, s \in M \quad (4.2)$$

$$\frac{dQ_k^s(t)}{dt} = \lambda_k^s + \sum_{j\in F}\sum_{l\in M} p_{j,k}\mu_j^{l,s}x_j^{l,s} - \sum_{l\in M}\mu_k^{s,l}x_k^{s,l} \quad \forall k \in F, s \in M \quad (4.3)$$

For notation, we let $(Q_k^s(t))' = \frac{dQ_k^s(t)}{dt}$. Put together, we can formulate an optimization model that maximizes the system throughput, which is our performance measure. Suppose we only produce one type of product, and let N be the number of steps required to produce this wafer. The throughput of the system up to time t is given by $\sum_{s\in M}\sum_{l\in M}\mu_N^{s,l}x_N^{s,l}t$. Noting that this is linear with respect to t, and t is non-negative, if we maximize $\sum_{s\in M}\sum_{l\in M}\mu_N^{s,l}x_N^{s,l}$, we are maximizing the throughput of the system.

Then, the optimization model to maximize the throughput (FM1) is

$$\max \sum_{s\in M}\sum_{l\in M} \mu_N^{s,l} x_N^{s,l}$$

$$\text{s.t.} \quad (Q_k^s(t))' \geq 0 \qquad \forall k \in F, s \in M : Q_k^s(0) = 0 \tag{4.4}$$

$$\sum_{k\in F}\sum_{l\in M} x_k^{s,l} \leq 1 \qquad \forall s \in M \tag{4.5}$$

$$0 \leq x_k^{s,l} \leq \alpha_k^{s,l} \qquad \forall k \in F, s, l \in M \tag{4.6}$$

Since $Q_k^s(t)$ are quantities on processing steps, constraint (4.4) states that for all the fluids that have zero inventory level at a given machine cannot decrease over time, otherwise the problem would be infeasible. Constraint (4.5) states that every machine cannot work more than 100% of the time. Constraint (4.6) defines the lower and upper bounds for each variable.

Once we obtain an optimal solution for the problem (FM1), for some pairs (k, s) we will have positive values of $(Q_k^s(t))'$, and negative values for others. For this reason, the solution obtained in (FM1) will be valid for a limited period of time τ as determined by the pair (k, s) which first reaches zero inventory,

$$\tau = \min \left\{ \frac{Q_k^s(0)}{\left|(Q_k^s(t))'\right|} : (Q_k^s(t))' < 0 \right\} \tag{4.7}$$

Therefore, the solution obtained in (FM1) will be valid for a period τ. After this period ends, we re-run the model, taking as input the state of the system at the end of the previous period. Doing this iteratively we obtain a set of solutions valid for its corresponding time period until we cover the time horizon.

We run our optimization model over fixed time interval periods, for instance, at the beginning of each day. Therefore, we impose that the solution delivered by the model has to be valid for at least the length of each period. For example, letting I be the length of each period, the fluid model with period length constraint (FM2) is

$$\max \sum_{s\in M}\sum_{l\in M} \mu_N^{s,l} x_N^{s,l}$$

$$\text{s.t.} \quad (Q_k^s(t))' \geq 0 \qquad \forall k \in F, s \in M : Q_k^s(0) = 0$$

$$\sum_{k\in F}\sum_{l\in M} x_k^{s,l} \leq 1 \qquad \forall s \in M$$

$$0 \leq x_k^{s,l} \leq \alpha_k^{s,l} \qquad \forall k \in F, s, l \in M$$

$$-\frac{(Q_k^s(t))'}{Q_k^s(0)} \leq z \qquad \forall k \in F, s \in M : Q_k^s(0) > 0 \tag{4.8}$$

$$z \leq \frac{1}{I} \tag{4.9}$$

We note that model (FM2) is the same as (FM1) but adds constraints (4.8) and (4.9). Constraint (4.8) ensures that the auxiliary variable z is always non-negative and greater or equal than $\frac{|(Q_k^s(t))'|}{Q_k^s(0)}$ for $(Q_k^s(t))' < 0$; therefore $\frac{1}{z} \leq \frac{Q_k^s(0)}{|(Q_k^s(t))'|}$. Constraint (4.9) states that $z \leq \frac{1}{I}$, and since z is non-negative because of (4.8), then $\frac{1}{z} \geq I$. Combining the two constraints yields $I \leq \frac{Q_k^s(0)}{|(Q_k^s(t))'|}$ for $(Q_k^s(t))' < 0$ which is what we want.

Translating the Fluid Model Solution into Decision Policies

As mentioned, we run our optimization model (FM2) at the beginning of a fixed time period to obtain a valid solution for the given period. But the solution given by the fluid model only specifies the proportion of time that we should work on a given fluid k in a given machine s, and does not specify the detailed job sequence at each machine. Therefore, we have to translate the solution given by the fluid model into a policy that tells us what job should be processed next.

To generate the policy, we adapt the proposal of [22]. We recall that we have to make two types of decisions: *Lot Dispatching* and *Lot Targeting*. We define three metrics, also referred to as gaps,

- Fluid-Machine-Machine Gap: Let $A_k^{s,l}(t)$ be the actual amount of fluid k processed in machine s and sent to machine l at time t, then

$$GAP_k^{s,l}(t) = \mu_k^{s,l} x_k^{s,l} t - A_k^{s,l}(t) \tag{4.10}$$

- Fluid-Machine Gap: Let $A_k^s(t)$ be the actual amount of fluid k processed at machine s up to time t, then

$$GAP_k^s(t) = \sum_{l \in M} \mu_k^{s,l} x_k^{s,l} t - A_k^s(t) \tag{4.11}$$

- Fluid Gap: Let $A_k(t)$ be the actual amount of fluid k processed up to time t, then

$$GAP_k(t) = \sum_{s \in M(k)} \sum_{l \in M} \mu_k^{s,l} x_k^{s,l} t - A_k(t) \tag{4.12}$$

Each time we need to make a decision, say at time t, we look at all the choices and pick the one with the biggest gap. In other words, we are trying to push the machine utilization on each fluid to its theoretical value. [68] use a similar approach to determine their dispatching policies, where they look at the Queuing Time Ratio (QTR) to decide which lot comes next, but in their case the policy is static, i.e., they only compute once the queuing time for each processing step, while we re-optimize at the beginning of each period, and therefore the solution adapts to the new status of the system.

4.3.3 Proposed Policies

The decision points for both *Lot Targeting* and *Lot Dispatching* occur when a machine finishes processing a wafer lot. At this time, we determine the target machine for the departing lot, and where to send it (to the target machine if available or to a storage location). After that, we decide which lot to pull from storage for processing at the current machine.

Lot Targeting

We divide *Lot Targeting* into two phases: *Lot Targeting phase 1* and *Lot Targeting phase 2* as follows:

- *Lot Targeting phase 1*: Once a machine finishes processing a given wafer, decide on which machine to send the wafer to for the next step of the processing.
- *Lot Targeting phase 2*: Once we decide the target machine, decide where to send the wafer (to that machine, if available, otherwise to a storage location).

Once a given machine s finishes processing fluid k, we have to decide the target machine and where to send the current lot. For instance, suppose that after fluid k is finished in machine s, it turns into fluid j at time t. We have two cases: either at least one machine in $M(j)$ is available or none is available. Clearly, when we have at least one machine available, both targeting phases rules are the same, but the rules are different when no machine in $M(j)$ is available.

- At least one machine is available: In this case, the target machine l^* is picked as follows

$$l^* = \operatorname{argmax} \left\{ GAP_k^{s,i}(t) : i \in M(j), i \text{ is available} \right\} \tag{4.13}$$

  Since machine l^* is available, we send the lot to machine l^*.

- No machines are available: In this case, the target machine is picked as follows

$$l^* = \operatorname{argmax} \left\{ GAP_k^{s,i}(t) : i \in M(j) \right\} \tag{4.14}$$

  Since machine l^* is not available, we send the lot to the closest available storage location to l^*.

Lot Dispatching and Lot Grouping

We propose to separate *Lot Dispatching* into two phases: *Lot Grouping* and *Lot Dispatching* as follows:

- *Lot Grouping*: Once a machine becomes available, choose a subset of all the lots waiting to be processed, not the actual lot to be processed.
- *Lot Dispatching*: Once a subset of lots has been selected, choose the actual lot to be processed.

The two-phase selection allows us to use the fluid model solution to choose the next fluid to be processed (the subset of all the lots in the queue), and to use commonly used rules to select the actual lot. This two-phase policy helped us to deal with the travel times issue, because we can include other conditions in the lot-selection policy, e.g., in the *Lot Grouping* phase we decide to pick lots that are in the location closest to the available machine, and then in the *Lot Dispatching* phase we choose the actual lot to be processed.

As mentioned, we use the the fluid model solution in the *Lot Grouping* phase to choose the next fluid to be processed. Suppose machine s becomes available at time t and has to pull a lot from a storage location. We have two cases: at least one fluid in the storage locations has machine s as target, or none of the fluids have machine s as target.

- At least one fluid has machine s as target: In this case, the picked fluid k^* is as follows:

$$k^* = \text{argmax}\left\{GAP_k^s(t) : k \in F(s), k \text{ has machine } s \text{ as target}\right\} \tag{4.15}$$

- No fluid has machine s as target: In this case, the picked fluid k^* is as follows:

$$k^* = \text{argmax}\left\{GAP_k(t) : k \in F(s)\right\} \tag{4.16}$$

Finally, having defined the fluid k^*, we pick the actual lot using the "First Come First Served" policy.

4.4 Results

We conduct simulations to evaluate the performance of the newly proposed rules, comparing our method with traditional targeting and dispatching rules. We use simple rules to compare our approach because they are used in practice, and the more sophisticated policies are difficult to implement and some of them focus on a different performance criteria than ours. Our primary interest is to understand how material handling, especially transfer delay, affects throughput. We consider a small size instance with varying travel time settings. We consider four cases: $i)$ no travel times, $ii)$ short travel times, $iii)$ medium travel times, and $iv)$ long travel times.

1. The effect of travel time

    - Travel time to processing time ratio (TR):

$$\text{TR} = \frac{\text{Average Travel time}}{\text{Average Processing time}} \tag{4.17}$$

2. Lot Targeting phase I rules

    - *Opt* : Use fluid model solution to pick the target machine.
    - *MW* : The target machine is the one with minimum workload.
    - *Closest* : The target machine is the closest to the current lot location.
    - *SPT* : The target machine is the one with shortest processing time.

3. Lot Grouping and Dispatching rules

    - *Opt FCFS* : In the *Lot Grouping* phase we use the fluid model solution to select the processing step. Then from all the lots in the selected processing step we use

a First Come First Serve (FCFS) policy to pick the actual lot to be processed.

- *Opt Closest* : In the *Lot Grouping* phase we use the fluid model to select the processing step, but only among lots in the closest storage location. Then we use a FCFS rule to decide the actual lot to be processed.
- *Opt SPT* : In the *Lot Grouping* phase we use the fluid model solution to select the processing step. Then from all the lots in the selected processing step we use SPT to pick the actual lot to be processed.
- *All FCFS* : We use a First Come First Serve policy to choose the lot to be processed.
- *All Closest* : We focus only on the closest storage location. Then we use FCFS to choose the actual lot to be processed.
- *All SPT* : We pick the lot with the Shortest Processing Time.

Note that policies All FCFS and All SPT do not do anything in the *Lot Grouping* phase. For the details when using the fluid model solution we refer the reader to sections 4.3.3 and 4.3.3.

4.4.1 Simulation Parameters and Assumptions

We assume that processing times and travel times have normal distribution with truncated minimum values in order to prevent too small values for designated mean times. The expected processing time is different for each machine and each processing step. Machine set-up times are not considered. Travel times among machines and storage locations are based on the bay in which each machine or storage location is located. When the transfer requests exceed the number of vehicles, the lot transfer time increases with the waiting time until a vehicle is allocated.

Our performance criterion is throughput. For each combination of dispatching and targeting rules, we conduct 10 independent replications to obtain a reliable estimate of our

stochastic simulation. The instance has 10 machines and 5 storage locations divided among 5 bays. Each lot completes 12 processing steps of 3 layers.

4.4.2 No Travel Times

In this case the Travel time to processing time ratio (TR) is 0. Figure 4.3 shows the results obtained for these instances. In the graph there are four groups of bars, each one corresponds to each one of the *Lot Targeting* rules. Within each group of bars we have the average throughput obtained using different *Lot Grouping* and *Lot Dispatching* policies. As we can see in Figure 4.3 the highest throughput were obtained when using the *Opt - Opt Closest* combination.

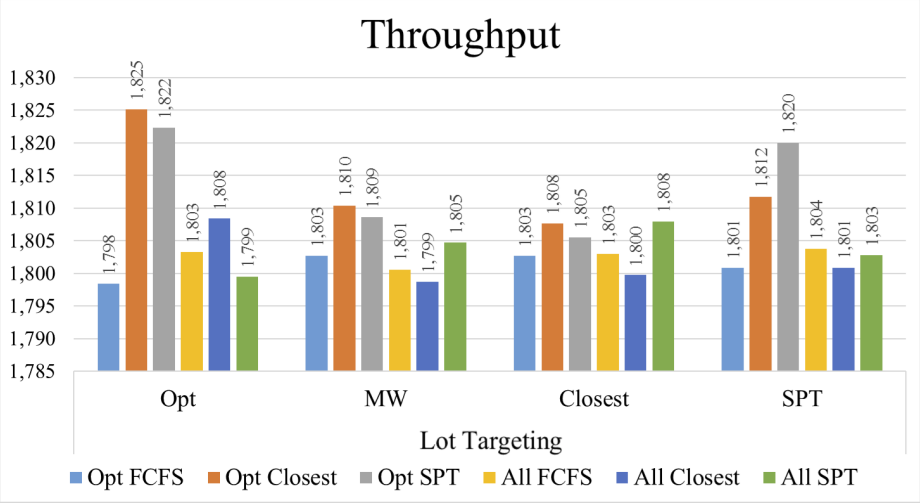


Figure 4.3: Total throughput, no travel times

4.4.3 Small Travel Times

In this case the TR is between 0.1 and 0.2. Figure 4.4 shows that the results are similar to the case without travel times, because even though we are considering travel times, the impact on the system's throughput of the travel times is minor compared to the impact of the processing times.

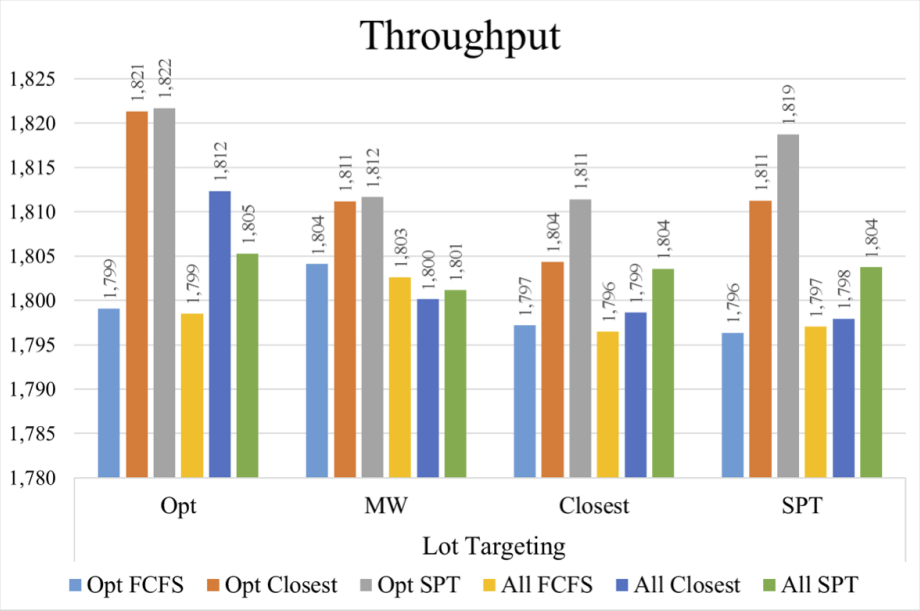


Figure 4.4: Total throughput, small travel times

4.4.4 Medium Travel Times

In this case TR is between 0.25 and 0.3, which we consider to be a more realistic scenario. Figure 4.5 shows that the best policy combination is to use *Opt* as the *Lot Targeting* policy, and either *Opt Closest* or *Opt SPT* as the *Lot Dispatching* policy. Furthermore, note that for every *Lot Targeting* policy, the optimization-based lot dispatching policy outperforms other dispatching policies, i.e., no matter what *Lot Targeting* policy we use, it is always better to choose an optimization-based dispatching policy.

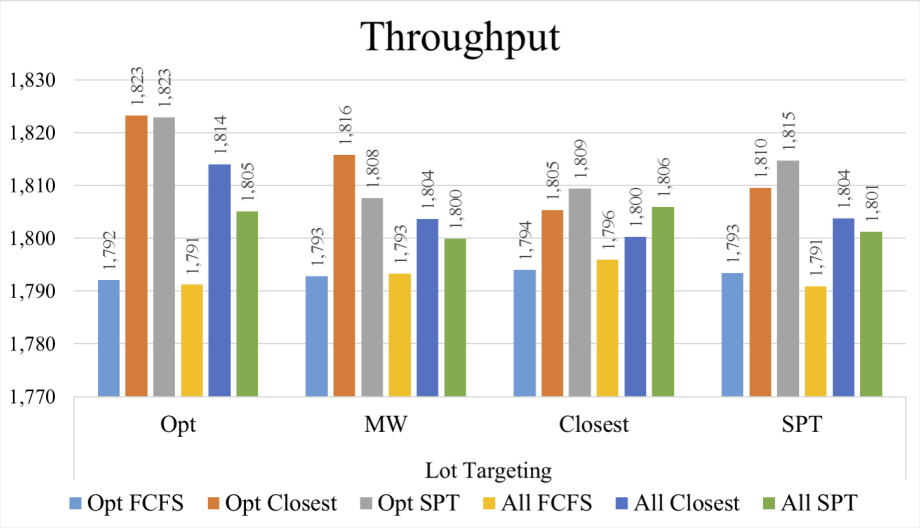


Figure 4.5: Total throughput, medium travel times

4.4.5 Long Travel Times

In this case TR is above 0.5. We note that the total throughput drops significantly. Figure 4.6 shows that while *Opt - Opt Closest* is not the best policy, it is better to use either *Closest - Opt Closest* or *Closest - All Closest* because the travel times become an important bottleneck and it is prudent to use policies that prioritize the selection of shorter travel times. This is an extreme case that does not happen in practice.

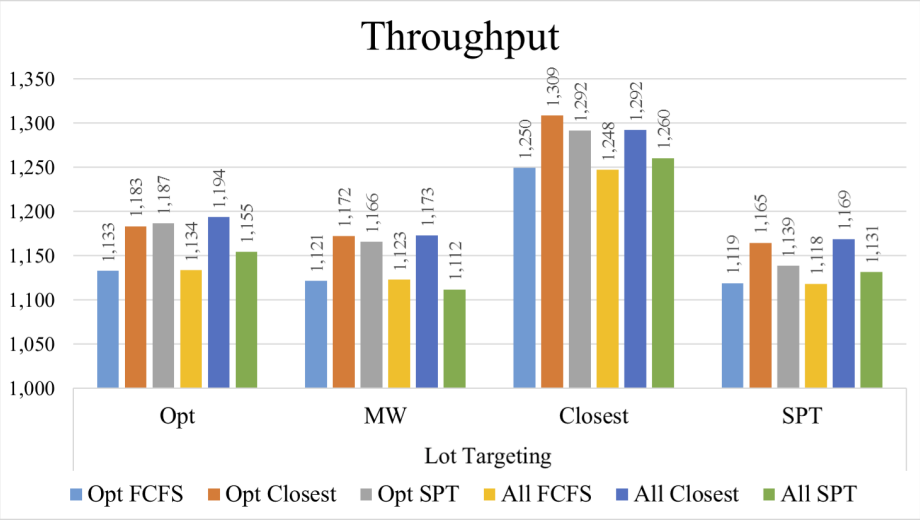


Figure 4.6: Total throughput, long travel times

4.4.6 Overall Analysis

As mentioned in Section 4.2, an important issue arises when a lot is dispatched to a storage location nearest its target machine, but instead is sent to a machine farther from that storage location. Figure 4.7 compares the target accuracy for the best *Lot Dispatching* policy under the two *Lot Grouping* policies. We consider the medium travel times case because it is the closest to reality. We can see that the optimization-based *Lot Dispatching* policy is always better, the difference is almost 20% in some cases. Recall that if the target machine is not available, we send the lot to its closest storage location. Hence, a higher target accuracy translates into less *long-distance storage-through transfers*, which is one of our main objectives.

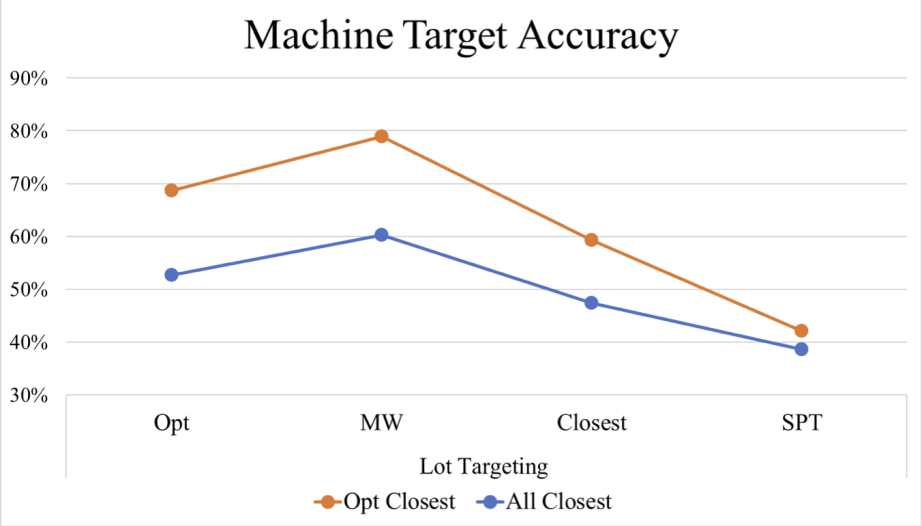


Figure 4.7: Machine target accuracy, medium travel time

Figure 4.8 compares the average machine utilization for the medium travel times case. Note that independently of the *Lot Targeting* policy, there is always an optimization-based *Lot Dispatching* policy that presents the highest machine utilization. For instance, when we use *Opt* as *Lot Targeting* policy, the *Opt SPT* shows the highest machine utilization, or when we use *MW* as *Lot Targeting* policy, the *Lot Dispatching* policy that presents the highest machine utilization is *Opt Closest*.

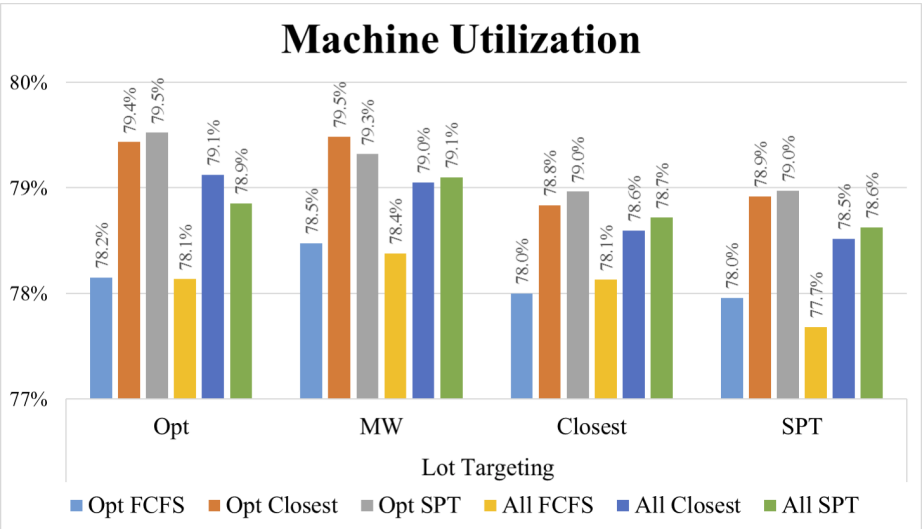


Figure 4.8: Machine utilization, medium travel time

Moreover, the bottleneck machine utilization is above 99% in all cases, and we found no difference among them in t-test statistics. Therefore, higher utilization in the rest of the machines can explain the better results in overall machine utilization obtained by the optimization solution. In all the other cases the machine target accuracy and machine utilization are better when we use the solutions obtained by the optimization model.

We also experimented with changing the number of machines, processing steps, processing and travel times, and facility configuration. The results obtained are similar to the instance described in this chapter. For every instance tested, the execution times for the optimization model are below 0.1 seconds. Since we propose a linear optimization model, we expect that the execution times for real world size instances would not increase drastically.

4.5 Conclusions

This chapter proposed a fluid model and algorithms for optimizing *Lot Targeting* and *Lot Dispatching* decision under the uncertainties of production. The main advantage of using the fluid model, instead of simple dispatching rules, is that its decisions are based on global information, i.e., the whole system's information, rather than local information, i.e., just machine level information. In simulation analyses, our approach improved production throughput except in instances with very long travel times. In all other cases, our approach not only outperforms other policies in throughput, it also improves machine target accuracy, in some cases over 20%, which results in a reduction of long distance transfers. In addition, the optimization-based approach leads to a higher machine utilization, not just in the bottleneck machines, which indicates a better use of limited resources. Furthermore, the low computation times for the optimization model make it practical for implementation.

Our current fluid model and simulation do not consider the time required to set-up a machine when changing from one product to another. Because set-up times are non-trivial, a next step is to incorporate set-up times in our *Lot Targeting* and *Lot Dispatching* policies either through inclusion in the fluid model directly or as an additional step in the targeting

and dispatching processes.

Appendices

APPENDIX A

OMITTED DEFINITIONS FROM CHAPTER 1

A.1 Network and graph definitions

An undirected graph is a pair $G = (V, E)$, where V is the set of vertices, or nodes, of the graph, and E is the set of edges (pair of nodes) of the graph. Likewise, a directed graph is a pair $D = (V, A)$, where V is the set of vertices, or nodes, of the graph, and A is the set of arcs (ordered pair of nodes) of the graph. If vertices $i, j \in V$ are connected by an edge $e \in E$ in graph G, we denote such an edge as $e = \{1, j\}$. Similarly, if there is an arc a connecting i and j in graph D, then we denote as $a = (i, j)$. Note that, edge $\{i, j\}$ can also be written as $\{j, i\}$, but arc (i, j) cannot be written as (j, i). In this thesis, we only work with simple and finite graphs, meaning that we do not consider cases with parallel edges, or arcs, and that the number of vertices is finite, which implies that the number of edges, or arcs, is finite as well. The undirected version of a directed graph $D = (V, A)$, is an undirected graph $G = (V, E)$, where we have an edge $\{i, j\} \in E$ if either $(i, j) \in A$ or $(j, i) \in A$. The edges of an undirected graph have a corresponding cost, denoted by c_e for all $e \in E$. Likewise, in a directed graph, the arcs have a corresponding cost, denoted by c_a for all $a \in A$. The tuple (V, E, c), or (V, A, c), is called a network.

For any $W \subseteq V$, we define the cut induced by W in G as $\delta(W) = \{\{i, j\} \in E : |e \cap W| = 1\}$. For the directed case, we make the distinction between $\delta^+(W) = \{(i, j) \in A : i \in W, j \in V \backslash W\}$, and $\delta^-(W) = \{(i, j) \in A : i \in V \backslash W, j \in W\}$. In the case that W is a singleton, i.e., $W = \{i\}$ we write $\delta(i)$ instead of $\delta(\{i\})$. In the undirected case, we define the degree of a vertex i as $|\delta(i)|$, which corresponds to the number of incident edges. In the directed case we define the indegree of vertex i as $|\delta^-(i)|$, and the outdegree of vertex i as $|\delta^+(i)|$.

In an undirected graph G, a path connecting i and j is defined as a sequence of edges that joins a set of distinct nodes, that include nodes i and j. More formally, a path P connecting i and j is a subgraph of G where $|\delta(i)| = |\delta(j)| = 1$, and the rest of nodes in P have degree 2. An undirected graph G is said to be connected if there is a path connecting every pair of nodes in G. A cycle C, is a connected subgraph of G, where all the nodes in C have degree 2. Note that in a path, the number of edges is the number of nodes minus 1, while in a cycle the number of nodes is equal to the number of edges. A tree T in G, is defined as an acyclic connected subgraph of G. Note that the number of edges in a tree is the number of nodes of the tree minus 1. The cost of a path is the sum of all the edges present in such a path, similarly we define the cost of a cycle and a tree.

In a directed graph D, an (i, j)-path is a sequence of arcs connecting nodes i and j. More formally, an (i, j)-path is a subgraph of D where i has outdegree 1 and indegree 0, j has outdegree 0 and indegree 1, and the rest of the nodes in the path have indegree and outdegree equal to 1. A directed graph $D = (V, A)$ is said to be strongly connected if for every pair of nodes $i, j \in V$ there is an (i, j)-path and a (j, i)-path in D. A directed cycle C, is a strongly connected subgraph of D, where all nodes have indegree and outdegree equal to 1. In an (i, j)-path, the number of arcs is the number of nodes minus 1, while in a directed cycle the number of nodes and arcs are equal. A directed graph D is said to be weakly connected if the undirected version of D is connected. A directed tree T in D, is defined as an acyclic weakly connected subgraph of D. An arborescence T rooted in r, also referred as r-arborescence, is a directed tree in D, where each node of T can be reached by r. The number of arcs in a directed tree is the number of nodes of the tree minus 1. The cost of an (i, j)-path is the sum of all the arcs present in such a directed path, similarly we define the cost of a directed cycle and a directed tree.

Given a set of commodities K, where each commodity $k \in K$ is determined by its source node $s_k \in V$, its sink node $t_k \in V$, and its demand $d_k \geq 0$; and a directed graph $D = (V, A)$, the fixed-charge multi-commodity network flow problem seeks to route every

commodity flow d_k, from s_k to t_k so the total cost is minimized. The total cost is the sum of the cost of each arc, where the cost of each arc a is the sum of a fixed cost, $c_a \geq 0$, derived from its use, and a variable cost proportional to the amount of flow sent through a. Moreover, each arc has a capacity denoted by $M_a \geq 0$.

APPENDIX B

SIMULATED ANNEALING INSTANCES

Table B.1 contains the information about all the instances studied in Chapter 3. Table B.1 contains the name of the instances, as well as the number of nodes, arcs and number of terminal nodes (excluding the root node). Column 'OPT' contains the value of an optimal solution to the instance. For instances whose optimal value is not known, we use the symbol '-'. Column 'BB' corresponds to the value of the best benchmark solution. Columns 'SA 1k' , 'SA-Test 1k' , and 'SA-Rect 1k' correspond to the values of the simulated annealing (SA), SA with solution improvement, and SA for rectilinear graphs using 1,000 iterations. Columns 'SA 5k' , 'SA-Test 5k' , and 'SA-Rect 5k' correspond to the values of the SA, SA with solution improvement, and SA for rectilinear graphs using 5,000 iterations. For not rectilinear instances, we use the symbol '-' in columns 'SA-Rect 1k' and 'SA-Rect 5k'.

Table B.1: Studied instances

Instance	$\|V\|$	$\|A\|$	$\|R\|$	OPT	BB	SA 1k	SA-Test 1k	SA-Rect 1k	SA 5k	SA-Test 5k	SA-Rect 5k
wrp3-11	128	454	10	1,100,361	1,100,374	1,100,361	1,100,361	-	1,100,361	1,100,361	-
wrp3-12	84	298	11	1,200,237	1,200,237	1,200,237	1,200,237	-	1,200,237	1,200,237	-
wrp3-13	311	1226	12	1,300,497	1,300,524	1,300,497	1,300,497	-	1,300,497	1,300,497	-
wrp3-14	128	494	13	1,400,250	1,400,251	1,400,250	1,400,250	-	1,400,250	1,400,250	-
wrp3-15	138	514	14	1,500,422	1,500,422	1,500,422	1,500,422	-	1,500,422	1,500,422	-
wrp3-16	204	748	15	1,600,208	1,600,224	1,600,208	1,600,208	-	1,600,208	1,600,208	-
wrp3-17	177	708	16	1,700,442	1,700,443	1,700,442	1,700,442	-	1,700,442	1,700,442	-
wrp3-19	189	706	18	1,900,439	1,900,448	1,900,441	1,900,439	-	1,900,439	1,900,439	-
wrp3-20	245	908	19	2,000,271	2,000,302	2,000,271	2,000,271	-	2,000,271	2,000,271	-
wrp3-21	237	888	20	2,100,522	2,100,529	2,100,525	2,100,522	-	2,100,522	2,100,522	-
wrp3-22	233	862	21	2,200,557	2,200,575	2,200,567	2,200,557	-	2,200,557	2,200,557	-
wrp3-23	132	460	22	2,300,245	2,300,245	2,300,252	2,300,245	-	2,300,245	2,300,245	-
wrp3-24	262	974	23	2,400,623	2,400,630	2,400,637	2,400,623	-	2,400,623	2,400,623	-
wrp3-25	246	936	24	2,500,540	2,500,563	2,500,557	2,500,540	-	2,500,540	2,500,540	-
wrp3-26	402	1560	25	2,600,484	2,600,499	2,600,494	2,600,484	-	2,600,484	2,600,484	-
wrp3-27	370	1442	26	2,700,502	2,700,514	2,700,528	2,700,502	-	2,700,502	2,700,502	-
wrp3-28	307	1118	27	2,800,379	2,800,396	2,800,433	2,800,379	-	2,800,379	2,800,379	-
wrp3-29	245	872	28	2,900,479	2,900,510	2,900,502	2,900,479	-	2,900,479	2,900,479	-
wrp3-30	467	1792	29	3,000,569	3,000,594	3,000,642	3,000,569	-	3,000,572	3,000,569	-
wrp3-31	323	1184	30	3,100,635	3,100,641	3,100,643	3,100,635	-	3,100,635	3,100,635	-
wrp3-33	437	1676	32	3,300,513	3,300,518	3,300,560	3,300,513	-	3,300,513	3,300,513	-
wrp3-34	1244	4948	33	3,400,646	3,400,696	3,400,749	3,400,646	-	3,400,648	3,400,646	-
wrp3-36	435	1636	35	3,600,610	3,600,642	3,600,722	3,600,610	-	3,600,612	3,600,610	-

wrp3-37	1011	4020	36	3,700,485	3,700,517	3,700,571	3,700,485	-	3,700,492	3,700,485	-
wrp3-38	603	2414	37	3,800,656	3,800,676	3,800,804	3,800,656	-	3,800,661	3,800,656	-
wrp3-39	703	3232	38	3,900,450	3,900,469	3,900,512	3,900,450	-	3,900,451	3,900,450	-
wrp3-41	178	614	40	4,100,466	4,100,467	4,100,566	4,100,466	-	4,100,471	4,100,466	-
wrp3-42	705	2746	41	4,200,598	4,200,634	4,200,752	4,200,598	-	4,200,607	4,200,598	-
wrp3-43	173	596	42	4,300,457	4,300,459	4,300,523	4,300,457	-	4,300,457	4,300,457	-
wrp3-45	1414	5626	44	4,500,860	4,500,915	4,501,149	4,500,861	-	4,500,892	4,500,860	-
wrp3-48	925	3476	47	4,800,552	4,800,584	4,800,790	4,800,553	-	4,800,580	4,800,552	-
wrp3-49	886	3600	48	4,900,882	4,900,914	4,901,181	4,900,882	-	4,900,928	4,900,882	-
wrp3-50	1119	4502	49	5,000,673	5,000,719	5,000,961	5,000,673	-	5,000,718	5,000,673	-
wrp3-52	701	2704	51	5,200,825	5,200,873	5,201,116	5,200,830	-	5,200,865	5,200,825	-
wrp3-53	775	2942	52	5,300,847	5,300,899	5,301,338	5,300,850	-	5,300,888	5,300,854	-
wrp3-55	1645	6372	54	5,500,888	5,500,950	5,501,238	5,500,888	-	5,500,926	5,500,888	-
wrp3-56	853	3180	55	5,600,872	5,600,930	5,601,250	5,600,877	-	5,600,919	5,600,872	-
wrp3-60	838	3526	59	6,001,164	6,001,173	6,001,541	6,001,164	-	6,001,201	6,001,164	-
wrp3-62	670	2632	61	6,201,016	6,201,057	6,201,359	6,201,016	-	6,201,101	6,201,016	-
wrp3-64	1822	7220	63	6,400,931	6,400,985	6,401,558	6,400,948	-	6,401,017	6,400,931	-
wrp3-66	2521	9716	65	6,600,922	6,600,997	6,601,439	6,600,941	-	6,601,017	6,600,922	-
wrp3-67	987	3846	66	6,700,776	6,700,820	6,701,134	6,700,782	-	6,700,831	6,700,777	-
wrp3-69	856	3242	68	6,900,841	6,900,879	6,901,393	6,900,841	-	6,900,971	6,900,841	-
wrp3-70	1468	5862	69	7,000,890	7,000,956	7,001,397	7,000,893	-	7,001,052	7,000,890	-
wrp3-71	1221	4828	70	7,101,028	7,101,118	7,101,764	7,101,041	-	7,101,240	7,101,028	-
wrp3-73	1890	7226	72	7,301,207	7,301,269	7,301,780	7,301,212	-	7,301,405	7,301,207	-
wrp3-74	1019	3882	73	7,400,759	7,400,789	7,401,302	7,400,769	-	7,400,850	7,400,763	-
wrp3-75	729	2790	74	7,501,020	7,501,030	7,501,678	7,501,020	-	7,501,150	7,501,020	-
wrp3-76	1761	6740	75	7,601,028	7,601,067	7,602,077	7,601,032	-	7,601,253	7,601,028	-
wrp3-78	2346	9312	77	7,801,094	7,801,190	7,802,045	7,801,115	-	7,801,411	7,801,098	-
wrp3-79	833	3190	78	7,900,444	7,900,486	7,900,831	7,900,448	-	7,900,525	7,900,444	-
wrp3-80	1491	5662	79	8,000,849	8,000,918	8,001,550	8,000,852	-	8,001,046	8,000,849	-
wrp3-83	3168	12440	82	8,300,906	8,300,966	8,301,849	8,300,926	-	8,301,150	8,300,918	-
wrp3-84	2356	9094	83	8,401,094	8,401,165	8,401,790	8,401,096	-	8,401,385	8,401,094	-
wrp3-85	528	2034	84	8,500,739	8,500,793	8,501,556	8,500,750	-	8,500,880	8,500,749	-
wrp3-86	1360	5214	85	86,000,746	86,000,818	86,001,855	86,000,749	-	86,000,999	86,000,746	-
wrp3-88	743	2818	87	88,001,175	88,001,186	88,002,079	88,001,175	-	88,001,552	88,001,175	-
wrp3-91	1343	5188	90	91,000,866	91,000,940	91,001,448	91,000,886	-	91,001,149	91,000,869	-
wrp3-92	1765	7226	91	92,000,764	92,000,825	92,001,246	92,000,768	-	92,001,042	92,000,766	-
wrp3-94	1976	7672	93	94,001,181	94,001,220	94,002,005	94,001,197	-	94,001,541	94,001,183	-
wrp3-96	2518	9970	95	96,001,172	96,001,311	96,001,949	96,001,195	-	96,001,645	96,001,174	-
wrp3-98	2265	9090	97	98,001,224	98,001,311	98,001,888	98,001,233	-	98,001,754	98,001,230	-
wrp3-99	2076	8144	98	99,001,097	99,001,201	99,001,889	99,001,117	-	99,001,624	99,001,099	-
wrp4-11	123	466	10	1,100,179	1,100,192	1,100,179	1,100,179	-	1,100,179	1,100,179	-
wrp4-13	110	376	12	1,300,798	1,300,806	1,300,798	1,300,798	-	1,300,798	1,300,798	-
wrp4-14	145	566	13	1,400,290	1,400,291	1,400,290	1,400,290	-	1,400,290	1,400,290	-
wrp4-15	193	738	14	1,500,405	1,500,408	1,500,405	1,500,405	-	1,500,405	1,500,405	-
wrp4-16	311	1158	15	1,601,190	1,601,237	1,601,190	1,601,190	-	1,601,190	1,601,190	-
wrp4-17	223	808	16	1,700,525	1,700,541	1,700,525	1,700,525	-	1,700,525	1,700,525	-
wrp4-18	211	760	17	1,801,464	1,801,499	1,801,464	1,801,464	-	1,801,464	1,801,464	-
wrp4-19	119	412	18	1,901,446	1,901,461	1,901,446	1,901,446	-	1,901,446	1,901,446	-
wrp4-21	529	2064	20	2,103,283	2,103,345	2,103,284	2,103,283	-	2,103,283	2,103,283	-
wrp4-22	294	1136	21	2,200,394	2,200,403	2,200,411	2,200,394	-	2,200,394	2,200,394	-
wrp4-23	257	1030	22	2,300,376	2,300,416	2,300,410	2,300,376	-	2,300,376	2,300,376	-
wrp4-24	493	1926	23	2,403,332	2,403,428	2,403,473	2,403,332	-	2,403,332	2,403,332	-
wrp4-25	422	1616	24	2,500,828	2,500,838	2,500,868	2,500,828	-	2,500,828	2,500,828	-
wrp4-26	396	1562	25	2,600,443	2,600,459	2,600,469	2,600,443	-	2,600,443	2,600,443	-
wrp4-27	243	994	26	2,700,441	2,700,464	2,700,473	2,700,441	-	2,700,441	2,700,441	-
wrp4-28	272	1090	27	2,800,466	2,800,488	2,800,470	2,800,466	-	2,800,466	2,800,466	-

wrp4-29	247	1010	28	2,900,484	2,900,518	2,900,513	2,900,484	-	2,900,486	2,900,484	-
wrp4-30	361	1448	29	3,000,526	3,000,563	3,000,586	3,000,527	-	3,000,530	3,000,526	-
wrp4-31	390	1572	30	3,100,526	3,100,553	3,100,576	3,100,526	-	3,100,526	3,100,526	-
wrp4-32	311	1264	31	3,200,554	3,200,626	3,200,640	3,200,554	-	3,200,554	3,200,554	-
wrp4-33	304	1142	32	3,300,655	3,300,677	3,300,743	3,300,655	-	3,300,656	3,300,655	-
wrp4-34	314	1300	33	3,400,525	3,400,572	3,400,634	3,400,532	-	3,400,539	3,400,525	-
wrp4-35	471	1908	34	3,500,601	3,500,672	3,500,698	3,500,612	-	3,500,616	3,500,601	-
wrp4-36	363	1500	35	3,600,596	3,600,638	3,600,672	3,600,596	-	3,600,612	3,600,596	-
wrp4-37	522	2108	36	3,700,647	3,700,709	3,700,811	3,700,651	-	3,700,651	3,700,647	-
wrp4-38	294	1236	37	3,800,606	3,800,632	3,800,740	3,800,606	-	3,800,620	3,800,606	-
wrp4-39	802	3106	38	3,903,734	3,903,854	3,904,607	3,903,734	-	3,903,806	3,903,734	-
wrp4-40	538	2176	39	4,000,758	4,000,793	4,000,931	4,000,767	-	4,000,764	4,000,758	-
wrp4-41	465	1910	40	4,100,695	4,100,752	4,100,860	4,100,695	-	4,100,706	4,100,695	-
wrp4-42	552	2262	41	4,200,701	4,200,715	4,200,931	4,200,721	-	4,200,724	4,200,701	-
wrp4-43	596	2296	42	4,301,508	4,301,605	4,301,895	4,301,508	-	4,301,536	4,301,508	-
wrp4-44	398	1576	43	4,401,504	4,401,577	4,402,007	4,401,504	-	4,401,524	4,401,504	-
wrp4-45	388	1630	44	4,500,728	4,500,817	4,500,998	4,500,745	-	4,500,781	4,500,742	-
wrp4-46	632	2574	45	4,600,756	4,600,812	4,601,058	4,600,772	-	4,600,795	4,600,756	-
wrp4-47	555	2196	46	4,701,318	4,701,434	4,701,884	4,701,318	-	4,701,350	4,701,318	-
wrp4-48	451	1650	47	4,802,220	4,802,305	4,803,377	4,802,220	-	4,802,334	4,802,220	-
wrp4-49	557	2160	48	4,901,968	4,902,073	4,902,856	4,901,968	-	4,902,009	4,901,968	-
wrp4-50	564	2224	49	5,001,625	5,001,697	5,002,201	5,001,625	-	5,001,687	5,001,625	-
wrp4-51	668	2612	50	5,101,616	5,101,738	5,102,081	5,101,616	-	5,101,712	5,101,616	-
wrp4-52	547	2230	51	5,201,081	5,201,255	5,201,494	5,201,081	-	5,201,139	5,201,081	-
wrp4-53	615	2464	52	5,301,351	5,301,395	5,302,217	5,301,351	-	5,301,406	5,301,351	-
wrp4-54	688	2776	53	5,401,534	5,401,644	5,402,504	5,401,550	-	5,401,600	5,401,534	-
wrp4-55	610	2402	54	5,501,952	5,502,057	5,502,658	5,501,953	-	5,502,007	5,501,952	-
wrp4-56	839	3234	55	5,602,299	5,602,412	5,603,612	5,602,311	-	5,602,375	5,602,299	-
wrp4-58	757	2986	57	5,801,466	5,801,588	5,802,021	5,801,480	-	5,801,609	5,801,466	-
wrp4-59	904	3612	58	5,901,592	5,901,667	5,902,401	5,901,597	-	5,901,773	5,901,592	-
wrp4-60	693	2740	59	6,001,782	6,001,889	6,002,447	6,001,792	-	6,001,999	6,001,782	-
wrp4-61	775	3076	60	6,102,210	6,102,361	6,103,257	6,102,210	-	6,102,443	6,102,210	-
wrp4-62	1283	4986	61	6,202,100	6,202,231	6,203,213	6,202,101	-	6,202,375	6,202,100	-
wrp4-63	1121	4454	62	6,301,479	6,301,646	6,302,331	6,301,482	-	6,301,649	6,301,480	-
wrp4-64	632	2562	63	6,401,996	6,402,099	6,403,005	6,402,012	-	6,402,199	6,402,002	-
wrp4-66	844	3382	65	6,602,931	6,603,034	6,604,878	6,602,938	-	6,603,146	6,602,931	-
wrp4-67	1518	6120	66	6,702,800	6,702,917	6,704,241	6,702,813	-	6,703,156	6,702,803	-
wrp4-68	917	3700	67	6,801,753	6,801,846	6,802,614	6,801,758	-	6,802,011	6,801,758	-
wrp4-69	574	2330	68	6,902,328	6,902,452	6,903,483	6,902,381	-	6,902,735	6,902,342	-
wrp4-70	637	2538	69	7,003,022	7,003,140	7,004,996	7,003,034	-	7,003,615	7,003,034	-
wrp4-71	802	3218	70	7,102,320	7,102,452	7,104,656	7,102,327	-	7,102,613	7,102,320	-
wrp4-72	1151	4548	71	7,202,807	7,202,974	7,204,473	7,202,819	-	7,203,096	7,202,810	-
wrp4-73	1898	7232	72	7,302,643	7,302,838	7,303,630	7,302,671	-	7,302,926	7,302,656	-
wrp4-74	802	3240	73	7,402,046	7,402,147	7,403,189	7,402,050	-	7,402,483	7,402,046	-
wrp4-75	938	3738	74	7,501,712	7,501,871	7,503,025	7,501,720	-	7,502,056	7,501,713	-
wrp4-76	766	3070	75	7,602,040	7,602,179	7,602,701	7,602,090	-	7,602,639	7,602,042	-
es10fst01	18	40	9	22,920,745	22,920,745	22,920,745	22,920,745	22,920,745	22,920,745	22,920,745	22,920,745
es10fst02	14	26	9	19,134,104	19,134,104	19,134,104	19,134,104	19,134,104	19,134,104	19,134,104	19,134,104
es10fst03	17	40	9	26,003,678	26,496,603	26,003,678	26,003,678	26,003,678	26,003,678	26,003,678	26,003,678
es10fst04	18	40	9	20,461,116	20,461,116	20,461,116	20,461,116	20,461,116	20,461,116	20,461,116	20,461,116
es10fst05	12	22	9	18,818,916	18,818,916	18,818,916	18,818,916	18,818,916	18,818,916	18,818,916	18,818,916
es10fst06	17	40	9	26,540,768	26,540,768	26,540,768	26,540,768	26,540,768	26,540,768	26,540,768	26,540,768
es10fst07	14	26	9	26,025,072	26,025,072	26,025,072	26,025,072	26,025,072	26,025,072	26,025,072	26,025,072
es10fst08	21	56	9	25,056,214	25,488,037	25,056,214	25,056,214	25,056,214	25,056,214	25,056,214	25,056,214
es10fst09	21	58	9	22,062,355	22,062,355	22,062,355	22,062,355	22,062,355	22,062,355	22,062,355	22,062,355
es10fst10	18	42	9	23,936,095	23,936,095	23,936,095	23,936,095	23,936,095	23,936,095	23,936,095	23,936,095

es10fst11	14	26	9	22,239,535	22,239,535	22,239,535	22,239,535	22,239,535	22,239,535	22,239,535	22,239,535
es10fst12	13	24	9	19,626,318	19,626,318	19,626,318	19,626,318	19,626,318	19,626,318	19,626,318	19,626,318
es10fst13	18	42	9	19,483,914	19,483,914	19,483,914	19,483,914	19,483,914	19,483,914	19,483,914	19,483,914
es10fst14	24	64	9	21,856,128	21,856,128	21,856,128	21,856,128	21,856,128	21,856,128	21,856,128	21,856,128
es10fst15	16	36	9	18,641,924	19,045,863	18,641,924	18,641,924	18,641,924	18,641,924	18,641,924	18,641,924
es20fst01	29	56	19	33,703,886	33,703,886	33,703,886	33,703,886	33,703,886	33,703,886	33,703,886	33,703,886
es20fst02	29	56	19	32,639,486	32,639,486	32,639,486	32,639,486	32,639,486	32,639,486	32,639,486	32,639,486
es20fst03	27	52	19	27,847,417	27,847,417	27,961,830	27,847,417	27,847,417	27,847,417	27,847,417	27,847,417
es20fst04	57	166	19	27,624,394	28,089,782	27,624,394	27,624,394	27,624,394	27,625,414	27,624,394	27,624,394
es20fst05	54	154	19	34,033,163	34,134,855	34,134,855	34,033,163	34,033,163	34,033,163	34,033,163	34,033,163
es20fst06	29	56	19	36,014,241	36,014,241	36,271,878	36,014,241	36,014,241	36,014,241	36,014,241	36,014,241
es20fst07	45	118	19	34,934,874	35,468,385	34,934,874	34,934,874	34,934,874	34,934,874	34,934,874	34,934,874
es20fst08	52	148	19	38,016,346	38,607,292	38,824,337	38,016,346	38,016,346	38,016,346	38,016,346	38,016,346
es20fst09	36	84	19	36,739,939	37,254,007	36,739,939	36,739,939	36,739,939	36,739,939	36,739,939	36,739,939
es20fst10	49	134	19	34,024,740	34,693,096	34,484,566	34,509,767	34,389,405	34,261,290	34,389,405	34,389,405
es20fst11	33	72	19	27,123,908	27,123,908	27,344,510	27,123,908	27,123,908	27,123,908	27,123,908	27,123,908
es20fst12	33	72	19	30,451,397	30,451,397	31,564,941	30,451,397	30,451,397	30,451,397	30,451,397	30,451,397
es20fst13	35	80	19	34,438,673	34,596,753	35,517,810	34,438,673	34,438,673	34,438,673	34,438,673	34,438,673
es20fst14	36	88	19	34,062,374	34,062,374	34,062,374	34,062,374	34,062,374	34,062,374	34,062,374	34,062,374
es20fst15	37	86	19	32,303,746	32,428,792	32,576,338	32,303,746	32,303,746	32,303,746	32,303,746	32,303,746
es30fst01	79	230	29	40,692,993	41,508,595	46,635,339	40,750,608	40,750,608	41,038,444	40,750,608	40,750,608
es30fst02	71	194	29	40,900,061	41,408,520	44,777,630	41,004,120	40,993,674	40,940,009	41,004,120	41,004,120
es30fst03	83	240	29	43,120,444	43,302,772	46,224,298	43,120,444	43,120,444	43,121,549	43,120,444	43,120,444
es30fst04	80	230	29	42,150,958	42,663,072	48,402,684	42,150,958	42,150,958	42,460,642	42,150,958	42,150,958
es30fst05	58	142	29	41,739,748	42,752,354	46,636,621	41,739,748	41,739,748	42,117,575	41,739,748	41,739,748
es30fst06	83	238	29	39,955,139	41,126,430	43,293,460	39,955,139	39,955,139	40,030,642	39,955,139	39,955,139
es30fst07	53	128	29	43,761,391	44,377,168	49,358,523	43,761,391	43,761,391	43,761,391	43,761,391	43,761,391
es30fst08	69	186	29	41,691,217	42,806,267	45,472,224	42,247,524	42,247,524	42,212,169	42,247,524	42,247,524
es30fst09	43	88	29	37,133,658	37,133,658	41,502,232	37,133,658	37,133,658	37,133,658	37,133,658	37,133,658
es30fst10	48	104	29	42,686,610	42,844,829	46,635,392	42,686,610	42,686,610	42,686,610	42,686,610	42,686,610
es30fst11	79	224	29	41,647,993	42,510,264	46,604,387	41,647,993	41,647,993	41,833,021	41,647,993	41,647,993
es30fst12	46	96	29	38,416,720	38,416,720	41,553,416	38,416,720	38,416,720	38,416,720	38,416,720	38,416,720
es30fst13	65	168	29	37,406,646	37,715,130	42,161,698	37,624,311	37,624,311	38,185,028	37,624,311	37,624,311
es30fst14	53	116	29	42,897,025	42,922,387	47,111,518	42,897,025	42,897,025	42,897,025	42,897,025	42,897,025
es30fst15	118	376	29	43,035,576	44,071,776	46,046,662	43,371,181	43,371,181	43,609,541	43,524,349	43,035,576
es40fst01	93	254	39	44,841,522	44,841,522	55,672,219	44,876,594	44,876,594	45,882,448	44,876,594	44,841,522
es40fst02	82	210	39	46,811,310	47,733,795	60,279,350	46,823,632	46,823,632	47,878,200	46,823,632	46,823,632
es40fst03	87	232	39	49,974,157	51,372,751	58,550,475	50,246,476	49,974,157	50,058,838	49,974,157	49,974,157
es40fst04	55	110	39	45,289,864	45,289,864	58,702,492	45,289,864	45,289,864	45,609,206	45,289,864	45,289,864
es40fst05	121	360	39	51,940,413	53,715,084	62,954,731	52,843,545	52,296,671	53,502,935	52,296,671	52,061,953
es40fst06	92	246	39	49,753,385	50,850,792	57,830,227	49,765,043	49,765,043	50,353,499	49,765,043	49,765,043
es40fst07	77	190	39	45,639,009	46,204,077	56,813,767	45,639,009	45,639,009	46,260,208	45,639,009	45,639,009
es40fst08	98	274	39	48,745,996	50,139,513	60,425,216	48,745,996	48,745,996	50,322,448	48,866,198	48,745,996
es40fst09	107	306	39	51,761,789	53,119,640	61,897,881	52,226,536	52,226,536	52,740,289	52,226,536	52,226,536
es40fst10	107	304	39	57,136,852	57,362,614	73,145,243	57,136,852	57,136,852	58,670,009	57,136,852	57,136,852
es40fst11	97	270	39	46,734,214	47,342,154	54,532,390	46,734,214	46,734,214	47,459,975	46,734,214	46,734,214
es40fst12	67	150	39	43,843,378	44,130,716	52,822,175	43,843,378	43,843,378	44,091,409	43,843,378	43,843,378
es40fst13	78	190	39	51,884,545	52,450,232	64,426,257	51,884,545	51,884,545	52,115,588	51,884,545	51,884,545
es40fst14	98	268	39	49,166,952	50,997,712	62,137,970	49,166,952	49,166,952	50,597,720	49,166,952	49,166,952
es40fst15	93	258	39	50,828,067	51,720,259	64,042,047	50,828,067	50,828,067	51,611,361	50,828,067	50,828,067
es50fst01	118	320	49	54,948,660	55,965,169	74,011,207	54,948,660	54,948,660	57,115,538	55,085,443	55,085,443
es50fst02	125	354	49	55,484,245	57,184,363	77,041,142	55,484,245	55,484,245	57,671,027	55,484,245	55,484,245
es50fst03	128	364	49	54,691,035	56,855,907	73,453,032	54,938,564	54,921,607	57,794,358	54,921,607	54,921,607
es50fst04	106	276	49	51,535,766	51,850,247	67,855,697	51,535,766	51,535,766	54,147,360	51,535,766	51,535,766
es50fst05	104	270	49	55,186,015	55,846,131	73,184,480	55,186,015	55,186,015	57,713,227	55,186,015	55,186,015
es50fst06	126	364	49	55,804,287	56,660,049	76,598,562	56,224,277	55,959,024	57,654,655	56,202,663	55,804,287

es50fst07	143	422	49	49,961,178	51,444,886	66,399,782	50,003,494	50,003,494	51,253,194	49,961,178	49,961,178
es50fst08	83	192	49	53,754,708	54,249,829	77,477,037	53,754,708	53,754,708	56,215,144	53,754,708	53,754,708
es50fst09	139	404	49	53,456,773	54,657,716	72,987,917	53,754,484	53,711,442	56,503,192	53,565,420	53,639,007
es50fst10	139	414	49	54,037,963	55,264,396	72,784,990	54,159,145	54,159,145	56,510,901	54,159,145	54,159,145
es50fst11	100	262	49	52,532,923	52,532,923	73,327,233	52,532,923	52,532,923	53,574,460	52,532,923	52,532,923
es50fst12	110	298	49	53,409,291	54,357,004	73,398,085	53,474,688	53,474,345	55,762,154	53,474,688	53,474,345
es50fst13	92	232	49	53,891,019	54,485,465	76,579,667	54,032,241	53,891,019	58,018,036	53,891,019	53,891,019
es50fst14	120	334	49	53,551,419	55,364,841	71,018,201	53,624,641	53,602,223	56,239,136	53,602,223	53,602,223
es50fst15	112	294	49	52,180,862	52,599,988	74,663,147	52,180,862	52,180,862	54,568,774	52,451,296	52,180,862
es60fst01	123	318	59	53,761,423	54,755,587	81,863,765	53,761,423	53,761,423	56,416,838	53,761,423	53,761,423
es60fst02	186	560	59	55,367,804	57,498,851	85,483,381	55,782,575	55,511,788	60,693,584	55,652,489	55,434,945
es60fst03	113	284	59	56,566,797	57,540,673	86,790,698	56,672,299	56,566,797	62,917,304	56,672,299	56,672,299
es60fst04	162	476	59	55,371,042	56,741,428	82,273,812	55,512,897	55,421,710	61,521,694	55,421,710	55,421,710
es60fst05	119	296	59	54,704,991	55,944,103	84,177,043	54,704,991	54,704,991	59,524,029	54,704,991	54,704,991
es60fst06	130	348	59	60,421,961	61,506,301	92,979,217	60,495,383	60,421,961	68,196,539	60,495,383	60,495,383
es60fst07	188	560	59	58,978,041	60,653,649	87,830,332	58,978,041	59,071,187	66,954,247	59,071,187	59,071,187
es60fst08	109	266	59	58,138,178	59,024,671	79,628,946	58,159,255	58,138,178	65,783,731	58,138,178	58,138,178
es60fst09	151	432	59	55,877,112	57,365,065	83,691,100	56,054,006	56,194,566	62,668,746	55,877,112	55,877,112
es60fst10	133	354	59	57,624,488	58,794,018	88,464,280	57,624,488	57,624,488	63,588,122	57,697,771	57,697,771
es60fst11	121	308	59	56,141,666	57,006,563	82,628,025	56,724,002	56,478,202	60,986,525	56,686,467	56,141,803
es60fst12	176	514	59	59,791,362	61,381,577	91,591,210	59,791,362	59,791,362	64,465,095	59,791,362	59,791,362
es60fst13	157	452	59	61,213,533	63,635,933	85,257,041	61,715,012	61,561,409	69,717,005	61,568,573	61,561,409
es60fst14	118	298	59	56,035,528	56,555,699	84,708,470	56,243,459	56,238,070	61,833,829	56,035,528	56,035,528
es60fst15	117	302	59	56,622,581	57,679,180	95,323,644	56,622,581	56,622,581	62,247,204	56,622,581	56,622,581
es70fst01	154	418	69	62,058,863	63,014,190	101,837,680	62,058,863	62,058,863	72,377,860	62,058,863	62,058,863
es70fst02	147	394	69	60,928,488	62,363,798	96,797,636	60,928,488	60,928,488	69,527,624	60,928,488	60,928,488
es70fst03	181	528	69	61,934,664	63,426,170	92,429,399	62,003,547	62,003,547	70,603,398	62,003,547	62,003,547
es70fst04	167	462	69	62,938,583	64,980,889	112,090,729	62,938,583	62,938,583	72,812,948	62,938,583	62,938,583
es70fst05	169	462	69	62,256,993	63,734,181	96,258,098	62,565,241	62,565,241	70,889,824	62,476,165	62,476,165
es70fst06	187	536	69	62,124,528	63,098,455	101,661,617	62,124,528	62,124,528	75,675,756	62,124,528	62,124,528
es70fst07	167	460	69	62,223,666	63,468,895	100,242,277	62,687,106	62,486,945	71,904,773	62,538,263	62,223,666
es70fst08	209	628	69	61,872,849	63,960,318	94,965,770	62,298,465	62,349,496	70,595,710	61,983,289	61,983,289
es70fst09	161	440	69	62,986,133	63,703,624	92,605,950	63,293,977	62,986,133	72,579,562	63,251,040	63,251,040
es70fst10	165	450	69	62,511,830	64,034,461	106,167,511	62,677,619	62,512,257	72,762,985	62,511,830	62,511,830
es70fst11	177	508	69	66,455,760	67,194,990	101,807,921	66,786,476	66,818,973	75,215,186	66,455,760	66,455,760
es70fst12	142	362	69	63,047,132	63,636,016	102,637,913	63,047,132	63,047,132	72,411,546	63,047,132	63,047,132
es70fst13	160	438	69	62,912,258	64,737,500	102,759,829	63,140,852	63,140,852	74,075,031	62,912,258	62,912,258
es70fst14	143	368	69	60,411,124	61,562,929	100,069,731	60,584,213	60,488,022	68,957,369	60,488,022	60,444,492
es70fst15	178	502	69	62,318,458	63,452,839	103,263,238	62,573,437	62,717,697	71,622,372	62,462,718	62,573,437
es80fst01	187	510	79	70,927,442	72,132,147	116,409,767	71,083,285	71,162,601	84,636,408	71,111,210	71,050,741
es80fst02	183	498	79	65,273,810	67,090,295	111,842,465	65,324,609	65,427,854	79,937,663	65,273,810	65,273,810
es80fst03	189	522	79	65,332,546	66,721,458	119,314,620	65,342,854	65,332,546	80,973,001	65,421,298	65,332,546
es80fst04	198	560	79	64,193,446	65,634,556	103,922,381	64,494,783	64,494,783	81,067,119	64,228,815	64,228,815
es80fst05	172	456	79	66,350,529	67,182,043	113,922,379	66,418,239	66,361,552	82,130,710	66,361,552	66,361,552
es80fst06	172	448	79	71,007,444	71,976,669	120,513,807	71,384,971	71,247,118	88,484,489	71,206,127	71,206,127
es80fst07	193	542	79	68,228,475	70,631,658	113,247,391	69,243,794	69,141,321	79,031,047	68,700,773	68,618,343
es80fst08	217	612	79	67,452,377	69,093,982	105,770,374	67,603,229	67,603,229	80,483,235	67,552,401	67,495,599
es80fst09	236	686	79	69,825,651	73,799,332	115,098,751	70,209,428	70,417,175	83,921,337	69,968,010	69,968,010
es80fst10	156	394	79	65,497,988	66,048,297	112,801,043	65,593,502	65,558,842	83,456,794	65,558,842	65,558,842
es80fst11	209	590	79	66,283,099	68,336,640	117,623,500	66,625,179	66,596,846	81,419,928	66,472,435	66,472,435
es80fst12	147	360	79	65,070,089	65,811,851	120,873,101	65,070,089	65,070,089	79,206,221	65,192,803	65,192,803
es80fst13	164	422	79	68,022,647	68,870,098	122,958,169	68,022,647	68,022,647	86,542,708	68,022,647	68,022,647
es80fst14	209	594	79	70,077,902	71,710,883	118,723,015	70,416,450	70,416,450	86,779,194	70,197,701	70,416,450
es80fst15	197	564	79	69,939,071	72,673,765	118,785,339	70,090,765	69,939,071	83,092,180	70,090,765	69,939,071
es90fst01	181	462	89	68,350,357	69,340,636	150,931,602	68,350,357	68,350,357	91,635,890	68,350,357	68,350,357
es90fst02	221	626	89	71,294,845	72,831,603	126,432,017	71,425,203	71,694,120	93,178,255	71,694,120	71,294,845

es90fst03	284	860	89	74,817,473	77,047,659	126,077,645	75,105,945	75,063,090	90,528,230	74,852,873	75,063,090
es90fst04	217	598	89	70,910,063	72,157,062	122,983,174	70,917,414	71,024,704	89,896,156	70,910,063	70,910,063
es90fst05	190	508	89	71,831,224	73,172,031	139,882,707	71,831,224	71,831,224	93,017,653	71,831,224	71,831,224
es90fst06	215	580	89	68,640,346	69,723,973	135,734,285	68,640,346	68,640,346	89,951,662	68,640,346	68,739,214
es90fst07	175	442	89	72,036,885	72,605,732	131,262,102	72,250,976	72,327,131	89,595,537	72,110,874	72,110,874
es90fst08	234	664	89	72,341,668	73,986,750	127,623,103	72,572,503	72,483,361	87,034,353	72,572,503	72,572,503
es90fst09	234	662	89	67,856,007	71,215,888	126,590,077	68,347,576	68,255,241	88,576,349	67,901,571	68,248,494
es90fst10	246	712	89	72,310,409	74,583,122	123,328,360	72,718,116	72,627,121	91,590,148	72,781,214	72,603,106
es90fst11	225	646	89	72,310,039	73,459,010	131,228,732	72,514,834	72,539,712	87,630,890	72,390,983	72,390,983
es90fst12	207	568	89	69,367,257	71,471,301	113,416,866	69,629,971	69,664,913	85,874,569	69,539,644	69,539,644
es90fst13	240	698	89	72,810,663	74,166,435	118,232,381	73,168,302	73,061,207	92,251,455	73,008,303	72,959,602
es90fst14	185	486	89	69,188,992	70,262,339	138,627,523	69,496,894	69,465,880	93,278,634	69,663,555	69,645,014
es90fst15	207	572	89	71,778,294	73,826,179	125,045,782	72,211,375	71,964,639	93,853,615	71,802,943	71,802,943
es100fst01	250	708	99	72,522,165	74,996,386	113,646,598	73,162,848	73,121,878	96,737,605	72,833,811	72,857,813
es100fst02	339	1044	99	75,176,630	76,681,625	131,323,543	75,815,432	75,569,313	99,417,305	75,340,116	75,567,309
es100fst03	189	466	99	72,746,006	73,849,547	123,680,439	72,902,717	73,187,062	100,744,772	72,902,717	72,902,717
es100fst04	188	470	99	74,342,392	75,693,600	161,542,475	74,693,569	74,579,736	101,101,314	74,579,736	74,579,736
es100fst05	188	476	99	75,670,198	76,897,439	152,269,080	75,938,104	75,846,518	105,331,568	75,932,463	75,932,463
es100fst06	301	904	99	74,414,990	76,297,034	131,514,016	74,873,914	74,774,210	98,133,810	74,488,218	74,788,218
es100fst07	276	802	99	77,740,576	80,265,546	137,610,940	78,404,419	78,617,102	101,561,113	78,521,419	78,318,231
es100fst08	210	552	99	73,033,178	74,839,270	121,805,114	73,525,091	73,519,753	92,789,504	73,519,753	73,519,753
es100fst09	248	684	99	77,952,027	80,059,348	137,043,274	79,275,897	79,014,040	101,586,959	78,836,698	78,865,289
es100fst10	229	624	99	75,952,202	77,683,619	129,303,378	76,702,908	76,634,779	101,901,491	76,486,096	76,375,025
es100fst11	253	724	99	78,674,859	80,283,998	136,353,831	78,968,098	78,848,637	102,233,108	78,848,637	78,848,637
es100fst12	266	770	99	76,131,099	78,387,376	153,396,885	76,564,015	76,689,893	103,218,548	76,301,181	76,282,932
es100fst13	254	722	99	74,604,990	76,822,679	152,766,695	75,196,104	74,844,199	99,738,847	74,988,766	74,858,345
es100fst14	198	506	99	78,632,795	79,774,666	149,842,856	78,632,795	78,632,795	108,976,665	78,632,795	78,632,795
es100fst15	231	638	99	70,446,493	72,805,295	127,537,710	70,798,205	70,699,648	93,858,665	70,652,858	70,573,355
diw0250	353	1216	10	350	355	350	350	350	350	350	350
diw0260	539	1970	11	468	468	468	468	468	468	468	468
diw0313	468	1644	13	397	397	397	397	397	397	397	397
diw0393	212	762	10	302	307	302	302	302	302	302	302
diw0445	1804	6622	32	1,363	1,388	1,445	1,363	1,363	1,370	1,363	1,363
diw0459	3636	13578	24	1,362	1,377	1,382	1,362	1,362	1,366	1,362	1,362
diw0460	339	1158	12	345	345	345	345	345	345	345	345
diw0473	2213	8270	24	1,098	1,110	1,105	1,098	1,098	1,098	1,098	1,098
diw0487	2414	8772	24	1,424	1,466	1,455	1,424	1,424	1,425	1,426	1,424
diw0495	938	3310	9	616	626	616	616	616	616	616	616
diw0513	918	3368	9	604	618	604	604	604	604	604	604
diw0523	1080	4030	9	561	561	561	561	561	561	561	561
diw0540	286	930	9	374	374	374	374	374	374	374	374
diw0559	3738	14026	17	1,570	1,613	1,570	1,570	1,570	1,570	1,570	1,570
diw0795	3221	11876	9	1,550	1,630	1,550	1,550	1,550	1,550	1,550	1,550
diw0801	3023	11150	9	1,587	1,609	1,587	1,587	1,587	1,587	1,587	1,587
dmxa0296	233	772	11	344	351	344	344	344	344	344	344
dmxa0296	233	772	11	344	351	344	344	344	344	344	344
dmxa0368	2050	7352	17	1,017	1,036	1,017	1,017	1,017	1,017	1,017	1,017
dmxa0454	1848	6572	15	914	958	914	914	914	914	914	914
dmxa0628	169	560	9	275	295	275	275	275	275	275	275
dmxa0734	663	2308	10	506	506	506	506	506	506	506	506
dmxa0848	499	1722	15	594	602	599	594	594	594	594	594
dmxa0903	632	2174	9	580	603	580	580	580	580	580	580
dmxa1109	343	1118	16	454	465	454	454	454	454	454	454
dmxa1200	770	2766	20	750	766	752	750	750	750	750	750
dmxa1304	298	1006	9	311	327	311	311	311	311	311	311
dmxa1516	720	2538	10	508	533	508	508	508	508	508	508

dmxa1721	1005	3462	17	780	789	780	780	780	780	780	780
dmxa1801	2333	8274	16	1,365	1,420	1,365	1,365	1,365	1,365	1,365	1,365
gap1307	342	1104	16	549	559	549	549	549	549	549	549
gap1413	541	1812	9	457	470	457	457	457	457	457	457
gap1500	220	748	16	254	254	254	254	254	254	254	254
gap1810	429	1404	16	482	487	482	482	482	482	482	482
gap1904	735	2512	20	763	778	765	763	763	763	763	763
gap2007	2039	7096	16	1,104	1,140	1,107	1,104	1,104	1,104	1,104	1,104
gap2119	1724	5950	28	1,244	1,307	1,336	1,244	1,244	1,244	1,244	1,244
gap2740	1196	4168	13	745	745	745	745	745	745	745	745
gap2800	386	1306	11	386	386	386	386	386	386	386	386
gap2975	179	586	9	245	245	245	245	245	245	245	245
gap3036	346	1166	12	457	473	457	457	457	457	457	457
gap3100	921	3116	10	640	682	640	640	640	640	640	640
msm0580	338	1082	10	467	490	467	467	467	467	467	467
msm0654	1290	4540	9	823	838	823	823	823	823	823	823
msm0709	1442	4806	15	884	884	884	884	884	884	884	884
msm0920	752	2528	25	806	826	834	806	806	806	806	806
msm1008	402	1390	10	494	518	494	494	494	494	494	494
msm1234	933	3264	12	550	563	550	550	550	550	550	550
msm1477	1199	4156	30	1,068	1,109	1,197	1,068	1,068	1,073	1,068	1,068
msm1707	278	956	10	564	574	564	564	564	564	564	564
msm1844	90	270	9	188	193	188	188	188	188	188	188
msm1931	875	3044	9	604	604	604	604	604	604	604	604
msm2000	898	3124	9	594	599	594	594	594	594	594	594
msm2152	2132	7404	36	1,590	1,649	1,782	1,590	1,590	1,604	1,590	1,590
msm2326	418	1446	13	399	409	399	399	399	399	399	399
msm2525	3031	10478	11	1,290	1,298	1,290	1,290	1,290	1,290	1,290	1,290
msm2601	2961	10200	15	1,440	1,458	1,440	1,440	1,440	1,440	1,440	1,440
msm2705	1359	4916	12	714	735	714	714	714	714	714	714
msm2802	1709	5926	17	926	951	926	926	926	926	926	926
msm2846	3263	11566	88	3,135	3,252	5,303	3,166	3,181	3,839	3,141	3,143
msm3277	1704	5982	11	869	869	869	869	869	869	869	869
msm3676	957	3108	9	607	607	607	607	607	607	607	607
msm4038	237	780	10	353	367	353	353	353	353	353	353
msm4114	402	1380	15	393	403	393	393	393	393	393	393
msm4190	391	1332	15	381	381	381	381	381	381	381	381
msm4224	191	604	10	311	315	311	311	311	311	311	311
msm4414	317	952	10	408	408	408	408	408	408	408	408
msm4515	777	2716	12	630	660	630	630	630	630	630	630
taq0023	572	1926	10	621	635	621	621	621	621	621	621
taq0431	1128	3810	12	897	907	897	897	897	897	897	897
taq0631	609	1864	9	581	591	581	581	581	581	581	581
taq0739	837	2876	15	848	877	848	848	848	848	848	848
taq0741	712	2434	15	847	897	847	847	847	847	847	847
taq0751	1051	3582	15	939	980	939	939	939	939	939	939
taq0891	331	1120	9	319	321	319	319	319	319	319	319
taq0910	310	1028	16	370	370	370	370	370	370	370	370
taq0920	122	388	16	210	210	210	210	210	210	210	210
taq0978	777	2478	9	566	571	566	566	566	566	566	566
b01	50	126	8	82	82	82	82	-	82	82	-
b02	50	126	12	83	83	83	83	-	83	83	-
b03	50	126	24	138	138	140	138	-	138	138	-
b04	50	200	8	59	59	59	59	-	59	59	-
b05	50	200	12	61	62	61	61	-	61	61	-
b06	50	200	24	122	123	125	122	-	122	122	-

b07	75	188	12	111	111	111	111	-	111	111	-
b08	75	188	18	104	104	104	104	-	104	104	-
b09	75	188	37	220	220	241	220	-	221	220	-
b10	75	300	12	86	90	86	86	-	86	86	-
b11	75	300	18	88	90	88	88	-	88	88	-
b12	75	300	37	174	174	194	174	-	174	174	-
b13	100	250	16	165	177	165	165	-	165	165	-
b14	100	250	24	235	236	235	235	-	235	235	-
b15	100	250	49	318	318	393	318	-	327	318	-
b16	100	400	16	127	133	127	127	-	127	127	-
b17	100	400	24	131	132	133	131	-	131	131	-
b18	100	400	49	218	222	263	218	-	222	218	-
i080-001	80	240	5	1,787	1,787	1,787	1,787	-	1,787	1,787	-
i080-002	80	240	5	1,607	1,607	1,607	1,607	-	1,607	1,607	-
i080-003	80	240	5	1,713	1,713	1,713	1,713	-	1,713	1,713	-
i080-004	80	240	5	1,866	1,866	1,866	1,866	-	1,866	1,866	-
i080-005	80	240	5	1,790	1,885	1,790	1,790	-	1,790	1,790	-
i080-011	80	700	5	1,479	1,488	1,479	1,479	-	1,479	1,479	-
i080-012	80	700	5	1,484	1,494	1,484	1,484	-	1,484	1,484	-
i080-013	80	700	5	1,381	1,383	1,381	1,381	-	1,381	1,381	-
i080-014	80	700	5	1,397	1,397	1,397	1,397	-	1,397	1,397	-
i080-015	80	700	5	1,495	1,496	1,495	1,495	-	1,495	1,495	-
i080-021	80	6320	5	1,175	1,175	1,175	1,175	-	1,175	1,175	-
i080-022	80	6320	5	1,178	1,178	1,178	1,178	-	1,178	1,178	-
i080-023	80	6320	5	1,174	1,174	1,174	1,174	-	1,174	1,174	-
i080-024	80	6320	5	1,161	1,161	1,161	1,161	-	1,161	1,161	-
i080-025	80	6320	5	1,162	1,162	1,162	1,162	-	1,162	1,162	-
i080-031	80	320	5	1,570	1,570	1,570	1,570	-	1,570	1,570	-
i080-032	80	320	5	2,088	2,089	2,088	2,088	-	2,088	2,088	-
i080-033	80	320	5	1,794	1,815	1,794	1,794	-	1,794	1,794	-
i080-034	80	320	5	1,688	1,771	1,688	1,688	-	1,688	1,688	-
i080-035	80	320	5	1,862	1,862	1,862	1,862	-	1,862	1,862	-
i080-041	80	1264	5	1,276	1,279	1,276	1,276	-	1,276	1,276	-
i080-042	80	1264	5	1,287	1,287	1,287	1,287	-	1,287	1,287	-
i080-043	80	1264	5	1,295	1,297	1,295	1,295	-	1,295	1,295	-
i080-044	80	1264	5	1,366	1,366	1,366	1,366	-	1,366	1,366	-
i080-045	80	1264	5	1,310	1,362	1,310	1,310	-	1,310	1,310	-
i080-101	80	240	7	2,608	2,704	2,608	2,608	-	2,608	2,608	-
i080-102	80	240	7	2,403	2,405	2,403	2,403	-	2,403	2,403	-
i080-103	80	240	7	2,603	2,672	2,603	2,603	-	2,603	2,603	-
i080-104	80	240	7	2,486	2,580	2,486	2,486	-	2,486	2,486	-
i080-105	80	240	7	2,203	2,203	2,203	2,203	-	2,203	2,203	-
i080-111	80	700	7	2,051	2,054	2,051	2,051	-	2,051	2,051	-
i080-112	80	700	7	1,885	1,893	1,885	1,885	-	1,885	1,885	-
i080-113	80	700	7	1,884	1,984	1,884	1,884	-	1,884	1,884	-
i080-114	80	700	7	1,895	1,895	1,895	1,895	-	1,895	1,895	-
i080-115	80	700	7	1,868	1,870	1,868	1,868	-	1,868	1,868	-
i080-121	80	6320	7	1,561	1,561	1,561	1,561	-	1,561	1,561	-
i080-122	80	6320	7	1,561	1,561	1,561	1,561	-	1,561	1,561	-
i080-123	80	6320	7	1,569	1,569	1,569	1,569	-	1,569	1,569	-
i080-124	80	6320	7	1,555	1,555	1,555	1,555	-	1,555	1,555	-
i080-125	80	6320	7	1,572	1,572	1,572	1,572	-	1,572	1,572	-
i080-131	80	320	7	2,284	2,371	2,284	2,284	-	2,284	2,284	-
i080-132	80	320	7	2,180	2,197	2,180	2,180	-	2,180	2,180	-
i080-133	80	320	7	2,261	2,261	2,261	2,261	-	2,261	2,261	-
i080-134	80	320	7	2,070	2,090	2,070	2,070	-	2,070	2,070	-

i080-135	80	320	7	2,102	2,102	2,105	2,102	-	2,102	2,102	-
i080-141	80	1264	7	1,788	1,823	1,788	1,788	-	1,788	1,788	-
i080-142	80	1264	7	1,708	1,708	1,708	1,708	-	1,708	1,708	-
i080-143	80	1264	7	1,767	1,777	1,767	1,767	-	1,767	1,767	-
i080-144	80	1264	7	1,772	1,772	1,772	1,772	-	1,772	1,772	-
i080-145	80	1264	7	1,762	1,762	1,762	1,762	-	1,762	1,762	-
i080-201	80	240	15	4,760	4,764	4,760	4,760	-	4,760	4,760	-
i080-202	80	240	15	4,650	4,680	4,650	4,650	-	4,652	4,650	-
i080-203	80	240	15	4,599	4,708	4,692	4,599	-	4,599	4,599	-
i080-204	80	240	15	4,492	4,583	4,499	4,499	-	4,499	4,499	-
i080-205	80	240	15	4,564	4,660	4,576	4,564	-	4,564	4,564	-
i080-211	80	700	15	3,631	3,746	3,631	3,631	-	3,631	3,631	-
i080-212	80	700	15	3,677	3,794	3,679	3,679	-	3,677	3,677	-
i080-213	80	700	15	3,678	3,751	3,730	3,693	-	3,730	3,678	-
i080-214	80	700	15	3,734	3,759	3,734	3,734	-	3,734	3,734	-
i080-215	80	700	15	3,681	3,767	3,681	3,681	-	3,681	3,681	-
i080-221	80	6320	15	3,158	3,158	3,158	3,158	-	3,158	3,158	-
i080-222	80	6320	15	3,141	3,141	3,141	3,141	-	3,141	3,141	-
i080-223	80	6320	15	3,156	3,156	3,156	3,156	-	3,156	3,156	-
i080-224	80	6320	15	3,159	3,159	3,159	3,159	-	3,159	3,159	-
i080-225	80	6320	15	3,150	3,150	3,150	3,150	-	3,150	3,150	-
i080-231	80	320	15	4,354	4,466	4,363	4,354	-	4,360	4,354	-
i080-232	80	320	15	4,199	4,392	4,290	4,199	-	4,199	4,199	-
i080-233	80	320	15	4,118	4,265	4,136	4,118	-	4,136	4,118	-
i080-234	80	320	15	4,274	4,274	4,302	4,274	-	4,274	4,274	-
i080-235	80	320	15	4,487	4,490	4,487	4,487	-	4,534	4,487	-
i080-241	80	1264	15	3,538	3,589	3,538	3,538	-	3,538	3,538	-
i080-242	80	1264	15	3,458	3,503	3,459	3,458	-	3,458	3,458	-
i080-243	80	1264	15	3,474	3,494	3,474	3,474	-	3,474	3,474	-
i080-244	80	1264	15	3,466	3,583	3,472	3,473	-	3,472	3,471	-
i080-245	80	1264	15	3,467	3,559	3,467	3,467	-	3,467	3,467	-
i080-301	80	240	19	5,519	5,628	5,519	5,519	-	5,519	5,519	-
i080-302	80	240	19	5,944	6,139	5,944	5,944	-	5,944	5,944	-
i080-303	80	240	19	5,777	5,978	5,788	5,852	-	5,777	5,780	-
i080-304	80	240	19	5,586	5,586	5,586	5,586	-	5,586	5,586	-
i080-305	80	240	19	5,932	5,966	5,936	5,932	-	5,932	5,932	-
i080-311	80	700	19	4,554	4,762	4,574	4,554	-	4,573	4,554	-
i080-312	80	700	19	4,534	4,659	4,563	4,534	-	4,534	4,534	-
i080-313	80	700	19	4,509	4,638	4,532	4,509	-	4,509	4,509	-
i080-314	80	700	19	4,515	4,554	4,515	4,515	-	4,515	4,515	-
i080-315	80	700	19	4,459	4,474	4,459	4,459	-	4,459	4,459	-
i080-321	80	6320	19	3,932	3,932	3,932	3,932	-	3,932	3,932	-
i080-322	80	6320	19	3,937	3,937	3,937	3,937	-	3,937	3,937	-
i080-323	80	6320	19	3,946	3,946	3,946	3,946	-	3,946	3,946	-
i080-324	80	6320	19	3,932	3,932	3,932	3,932	-	3,932	3,932	-
i080-325	80	6320	19	3,924	3,924	3,924	3,924	-	3,924	3,924	-
i080-331	80	320	19	5,226	5,334	5,301	5,230	-	5,226	5,226	-
i080-332	80	320	19	5,362	5,388	5,378	5,362	-	5,362	5,362	-
i080-333	80	320	19	5,381	5,467	5,441	5,441	-	5,441	5,441	-
i080-334	80	320	19	5,264	5,269	5,270	5,265	-	5,267	5,267	-
i080-335	80	320	19	4,953	5,059	5,038	4,953	-	4,953	4,953	-
i080-341	80	1264	19	4,236	4,268	4,236	4,265	-	4,236	4,236	-
i080-342	80	1264	19	4,337	4,375	4,345	4,342	-	4,342	4,337	-
i080-343	80	1264	19	4,246	4,371	4,246	4,246	-	4,246	4,246	-
i080-344	80	1264	19	4,310	4,405	4,310	4,310	-	4,310	4,310	-
i080-345	80	1264	19	4,341	4,451	4,391	4,351	-	4,403	4,341	-

i160-001	160	480	6	2,490	2,513	2,490	2,490	-	2,490	2,490	-
i160-002	160	480	6	2,158	2,160	2,158	2,158	-	2,158	2,158	-
i160-003	160	480	6	2,297	2,297	2,297	2,297	-	2,297	2,297	-
i160-004	160	480	6	2,370	2,495	2,370	2,370	-	2,370	2,370	-
i160-005	160	480	6	2,495	2,594	2,495	2,495	-	2,495	2,495	-
i160-011	160	1624	6	1,677	1,677	1,677	1,677	-	1,677	1,677	-
i160-012	160	1624	6	1,750	1,774	1,750	1,750	-	1,750	1,750	-
i160-013	160	1624	6	1,661	1,759	1,661	1,661	-	1,661	1,661	-
i160-014	160	1624	6	1,778	1,796	1,778	1,778	-	1,778	1,778	-
i160-015	160	1624	6	1,768	1,768	1,768	1,768	-	1,768	1,768	-
i160-021	160	25440	6	1,352	1,352	1,352	1,352	-	1,352	1,352	-
i160-022	160	25440	6	1,365	1,365	1,365	1,365	-	1,365	1,365	-
i160-023	160	25440	6	1,351	1,351	1,351	1,351	-	1,351	1,351	-
i160-024	160	25440	6	1,371	1,371	1,371	1,371	-	1,371	1,371	-
i160-025	160	25440	6	1,366	1,366	1,366	1,366	-	1,366	1,366	-
i160-031	160	640	6	2,170	2,170	2,170	2,170	-	2,170	2,170	-
i160-032	160	640	6	2,330	2,425	2,330	2,330	-	2,330	2,330	-
i160-033	160	640	6	2,101	2,107	2,101	2,101	-	2,101	2,101	-
i160-034	160	640	6	2,083	2,083	2,083	2,083	-	2,083	2,083	-
i160-035	160	640	6	2,103	2,103	2,103	2,103	-	2,103	2,103	-
i160-041	160	5088	6	1,494	1,543	1,494	1,494	-	1,494	1,494	-
i160-042	160	5088	6	1,486	1,486	1,486	1,486	-	1,486	1,486	-
i160-043	160	5088	6	1,549	1,561	1,549	1,549	-	1,549	1,549	-
i160-044	160	5088	6	1,478	1,478	1,478	1,478	-	1,478	1,478	-
i160-045	160	5088	6	1,554	1,560	1,554	1,554	-	1,554	1,554	-
i160-101	160	480	11	3,859	3,859	3,859	3,859	-	3,859	3,859	-
i160-102	160	480	11	3,747	3,824	3,747	3,747	-	3,747	3,747	-
i160-103	160	480	11	3,837	3,837	3,837	3,837	-	3,837	3,837	-
i160-104	160	480	11	4,063	4,065	4,063	4,063	-	4,063	4,063	-
i160-105	160	480	11	3,563	3,655	3,563	3,563	-	3,563	3,563	-
i160-111	160	1624	11	2,869	2,869	2,869	2,869	-	2,869	2,869	-
i160-112	160	1624	11	2,924	2,934	2,924	2,924	-	2,924	2,924	-
i160-113	160	1624	11	2,866	2,898	2,866	2,866	-	2,866	2,866	-
i160-114	160	1624	11	2,989	3,047	3,010	3,010	-	3,024	3,024	-
i160-115	160	1624	11	2,937	2,944	2,937	2,937	-	2,937	2,937	-
i160-121	160	25440	11	2,363	2,363	2,363	2,363	-	2,363	2,363	-
i160-122	160	25440	11	2,348	2,348	2,348	2,348	-	2,348	2,348	-
i160-123	160	25440	11	2,355	2,355	2,355	2,355	-	2,355	2,355	-
i160-124	160	25440	11	2,352	2,352	2,352	2,352	-	2,352	2,352	-
i160-125	160	25440	11	2,351	2,351	2,351	2,351	-	2,351	2,351	-
i160-131	160	640	11	3,356	3,431	3,356	3,356	-	3,356	3,356	-
i160-132	160	640	11	3,450	3,576	3,450	3,450	-	3,450	3,450	-
i160-133	160	640	11	3,585	3,763	3,585	3,585	-	3,585	3,585	-
i160-134	160	640	11	3,470	3,489	3,470	3,470	-	3,470	3,470	-
i160-135	160	640	11	3,716	3,732	3,716	3,716	-	3,716	3,716	-
i160-141	160	5088	11	2,549	2,650	2,549	2,549	-	2,549	2,549	-
i160-142	160	5088	11	2,562	2,634	2,562	2,571	-	2,571	2,562	-
i160-143	160	5088	11	2,557	2,663	2,557	2,557	-	2,557	2,557	-
i160-144	160	5088	11	2,607	2,612	2,610	2,607	-	2,607	2,607	-
i160-145	160	5088	11	2,578	2,578	2,578	2,578	-	2,578	2,578	-
i160-201	160	480	23	6,923	7,200	6,927	6,923	-	6,923	6,923	-
i160-202	160	480	23	6,930	7,038	6,930	6,930	-	6,934	6,930	-
i160-203	160	480	23	7,243	7,330	7,337	7,243	-	7,252	7,243	-
i160-204	160	480	23	7,068	7,273	7,149	7,077	-	7,068	7,068	-
i160-205	160	480	23	7,122	7,323	7,214	7,132	-	7,122	7,122	-
i160-211	160	1624	23	5,583	5,650	5,613	5,640	-	5,670	5,648	-

i160-212	160	1624	23	5,643	5,839	5,739	5,649	-	5,725	5,652	-
i160-213	160	1624	23	5,647	5,743	5,724	5,681	-	5,720	5,686	-
i160-214	160	1624	23	5,720	5,769	5,857	5,808	-	5,808	5,734	-
i160-215	160	1624	23	5,518	5,849	5,633	5,603	-	5,633	5,597	-
i160-221	160	25440	23	4,729	4,729	4,729	4,729	-	4,729	4,729	-
i160-222	160	25440	23	4,697	4,697	4,697	4,697	-	4,697	4,697	-
i160-223	160	25440	23	4,730	4,730	4,730	4,730	-	4,730	4,730	-
i160-224	160	25440	23	4,721	4,721	4,721	4,721	-	4,721	4,721	-
i160-225	160	25440	23	4,728	4,728	4,728	4,728	-	4,728	4,728	-
i160-231	160	640	23	6,662	6,742	6,761	6,725	-	6,726	6,732	-
i160-232	160	640	23	6,558	6,842	6,639	6,558	-	6,566	6,566	-
i160-233	160	640	23	6,339	6,427	6,500	6,339	-	6,339	6,339	-
i160-234	160	640	23	6,594	6,610	6,675	6,594	-	6,594	6,594	-
i160-235	160	640	23	6,764	6,930	6,976	6,767	-	6,846	6,846	-
i160-241	160	5088	23	5,086	5,145	5,107	5,086	-	5,086	5,086	-
i160-242	160	5088	23	5,106	5,251	5,142	5,150	-	5,139	5,106	-
i160-243	160	5088	23	5,050	5,165	5,094	5,095	-	5,095	5,051	-
i160-244	160	5088	23	5,076	5,212	5,141	5,139	-	5,118	5,114	-
i160-245	160	5088	23	5,084	5,167	5,094	5,098	-	5,098	5,084	-
i160-301	160	480	39	11,816	12,025	12,986	11,905	-	11,918	11,904	-
i160-302	160	480	39	11,497	11,640	12,558	11,591	-	11,686	11,497	-
i160-303	160	480	39	11,445	11,553	12,326	11,445	-	11,540	11,445	-
i160-304	160	480	39	11,448	11,542	12,138	11,521	-	11,546	11,521	-
i160-305	160	480	39	11,423	11,520	12,065	11,465	-	11,583	11,456	-
i160-311	160	1624	39	9,135	9,242	9,463	9,256	-	9,355	9,255	-
i160-312	160	1624	39	9,052	9,288	9,454	9,226	-	9,197	9,223	-
i160-313	160	1624	39	9,159	9,209	9,578	9,369	-	9,431	9,309	-
i160-314	160	1624	39	8,941	8,958	9,359	9,065	-	9,045	9,057	-
i160-315	160	1624	39	9,086	9,225	9,362	9,154	-	9,161	9,143	-
i160-321	160	25440	39	7,876	7,903	7,903	7,903	-	7,903	7,876	-
i160-322	160	25440	39	7,859	7,892	7,892	7,892	-	7,892	7,859	-
i160-323	160	25440	39	7,876	7,883	7,883	7,883	-	7,883	7,883	-
i160-324	160	25440	39	7,884	7,912	7,912	7,884	-	7,912	7,884	-
i160-325	160	25440	39	7,862	7,915	7,877	7,877	-	7,877	7,877	-
i160-331	160	640	39	10,414	10,614	11,015	10,505	-	10,583	10,489	-
i160-332	160	640	39	10,806	10,845	11,605	10,871	-	10,836	10,806	-
i160-333	160	640	39	10,561	10,651	11,220	10,637	-	10,668	10,624	-
i160-334	160	640	39	10,327	10,697	10,928	10,489	-	10,590	10,392	-
i160-335	160	640	39	10,589	10,730	11,192	10,772	-	10,672	10,594	-
i160-341	160	5088	39	8,331	8,427	8,433	8,404	-	8,412	8,397	-
i160-342	160	5088	39	8,348	8,518	8,538	8,370	-	8,489	8,384	-
i160-343	160	5088	39	8,275	8,318	8,354	8,342	-	8,358	8,352	-
i160-344	160	5088	39	8,307	8,363	8,389	8,324	-	8,363	8,324	-
i160-345	160	5088	39	8,327	8,441	8,434	8,382	-	8,436	8,380	-
alue2087	1244	3942	33	1,049	1,055	1,189	1,052	1,049	1,052	1,052	1,052
alue2105	1220	3716	33	1,032	1,051	1,105	1,032	1,032	1,045	1,032	1,032
alue3146	3626	11738	63	2,240	2,316	3,130	2,254	2,245	2,408	2,240	2,243
alue5067	3524	11120	67	2,586	2,688	3,902	2,608	2,613	2,934	2,586	2,586
alue6179	3372	10426	66	2,452	2,483	3,674	2,465	2,475	2,679	2,452	2,454
alue6951	2818	8838	66	2,386	2,519	3,604	2,414	2,413	2,674	2,386	2,387
alue7229	940	2948	33	824	826	930	824	824	842	824	824
alut0787	1160	4178	33	982	989	1,050	982	982	988	982	982
alut0805	966	3332	33	958	966	1,094	958	958	958	958	958
alut1181	3041	11386	63	2,353	2,449	3,569	2,381	2,381	2,592	2,358	2,355
alut2764	387	1252	33	640	650	723	640	640	640	640	640
i320-001	320	960	7	2,672	2,672	2,672	2,672	-	2,672	2,672	-

i320-002	320	960	7	2,847	2,847	2,847	2,847	-	2,847	2,847	-
i320-003	320	960	7	2,972	2,984	2,972	2,972	-	2,972	2,972	-
i320-004	320	960	7	2,905	2,989	2,905	2,905	-	2,905	2,905	-
i320-005	320	960	7	2,991	3,093	2,991	2,991	-	2,991	2,991	-
i320-011	320	3690	7	2,053	2,061	2,053	2,053	-	2,053	2,053	-
i320-012	320	3690	7	1,997	2,062	1,997	1,997	-	1,997	1,997	-
i320-013	320	3690	7	2,072	2,072	2,072	2,072	-	2,072	2,072	-
i320-014	320	3690	7	2,061	2,073	2,061	2,061	-	2,061	2,061	-
i320-015	320	3690	7	2,059	2,070	2,059	2,059	-	2,059	2,059	-
i320-021	320	102080	7	1,553	1,553	1,553	1,553	-	1,553	1,553	-
i320-022	320	102080	7	1,565	1,565	1,565	1,565	-	1,565	1,565	-
i320-023	320	102080	7	1,549	1,549	1,549	1,549	-	1,549	1,549	-
i320-024	320	102080	7	1,553	1,553	1,553	1,553	-	1,553	1,553	-
i320-025	320	102080	7	1,550	1,550	1,550	1,550	-	1,550	1,550	-
i320-031	320	1280	7	2,673	2,764	2,673	2,673	-	2,673	2,673	-
i320-032	320	1280	7	2,770	2,982	2,770	2,770	-	2,770	2,770	-
i320-033	320	1280	7	2,769	2,865	2,769	2,769	-	2,769	2,769	-
i320-034	320	1280	7	2,521	2,529	2,521	2,521	-	2,521	2,521	-
i320-035	320	1280	7	2,385	2,656	2,385	2,385	-	2,385	2,385	-
i320-041	320	20416	7	1,707	1,761	1,707	1,707	-	1,707	1,707	-
i320-042	320	20416	7	1,682	1,686	1,682	1,682	-	1,682	1,682	-
i320-043	320	20416	7	1,723	1,760	1,723	1,723	-	1,723	1,723	-
i320-044	320	20416	7	1,681	1,770	1,681	1,681	-	1,681	1,681	-
i320-045	320	20416	7	1,686	1,686	1,686	1,686	-	1,686	1,686	-
i320-101	320	960	16	5,548	5,554	5,548	5,548	-	5,548	5,548	-
i320-102	320	960	16	5,556	5,754	5,556	5,556	-	5,556	5,556	-
i320-103	320	960	16	6,239	6,448	6,239	6,239	-	6,243	6,239	-
i320-104	320	960	16	5,703	6,057	5,797	5,780	-	5,703	5,703	-
i320-105	320	960	16	5,928	6,139	5,928	5,932	-	5,928	5,936	-
i320-111	320	3690	16	4,273	4,370	4,283	4,286	-	4,278	4,276	-
i320-112	320	3690	16	4,213	4,328	4,213	4,213	-	4,213	4,213	-
i320-113	320	3690	16	4,205	4,328	4,205	4,205	-	4,212	4,205	-
i320-114	320	3690	16	4,104	4,215	4,104	4,104	-	4,142	4,104	-
i320-115	320	3690	16	4,238	4,261	4,317	4,238	-	4,314	4,238	-
i320-121	320	102080	16	3,321	3,321	3,321	3,321	-	3,321	3,321	-
i320-122	320	102080	16	3,314	3,314	3,314	3,314	-	3,314	3,314	-
i320-123	320	102080	16	3,332	3,332	3,332	3,332	-	3,332	3,332	-
i320-124	320	102080	16	3,323	3,323	3,323	3,323	-	3,323	3,323	-
i320-125	320	102080	16	3,340	3,345	3,340	3,340	-	3,340	3,340	-
i320-131	320	1280	16	5,255	5,388	5,255	5,268	-	5,255	5,255	-
i320-132	320	1280	16	5,052	5,264	5,058	5,052	-	5,060	5,052	-
i320-133	320	1280	16	5,125	5,237	5,125	5,125	-	5,125	5,125	-
i320-134	320	1280	16	5,272	5,301	5,295	5,272	-	5,335	5,272	-
i320-135	320	1280	16	5,342	5,447	5,367	5,342	-	5,355	5,342	-
i320-141	320	20416	16	3,606	3,638	3,611	3,606	-	3,610	3,606	-
i320-142	320	20416	16	3,567	3,590	3,567	3,567	-	3,567	3,567	-
i320-143	320	20416	16	3,561	3,648	3,561	3,561	-	3,561	3,561	-
i320-144	320	20416	16	3,512	3,648	3,512	3,512	-	3,512	3,512	-
i320-145	320	20416	16	3,601	3,610	3,601	3,601	-	3,601	3,601	-
i320-201	320	960	33	10,044	10,344	10,650	10,044	-	10,189	10,044	-
i320-202	320	960	33	11,223	11,254	11,731	11,231	-	11,235	11,233	-
i320-203	320	960	33	10,148	10,520	10,838	10,227	-	10,161	10,148	-
i320-204	320	960	33	10,275	10,464	11,015	10,367	-	10,429	10,361	-
i320-205	320	960	33	10,573	10,797	10,987	10,725	-	10,651	10,642	-
i320-211	320	3690	33	8,039	8,299	8,488	8,157	-	8,172	8,116	-
i320-212	320	3690	33	8,044	8,231	8,375	8,140	-	8,184	8,143	-

i320-213	320	3690	33	7,984	8,080	8,291	8,222	-	8,228	8,155	-
i320-214	320	3690	33	8,046	8,191	8,399	8,112	-	8,193	8,051	-
i320-215	320	3690	33	8,015	8,202	8,425	8,139	-	8,177	8,041	-
i320-221	320	102080	33	6,679	6,700	6,697	6,697	-	6,697	6,697	-
i320-222	320	102080	33	6,686	6,688	6,688	6,688	-	6,688	6,688	-
i320-223	320	102080	33	6,695	6,709	6,709	6,709	-	6,709	6,709	-
i320-224	320	102080	33	6,694	6,694	6,694	6,694	-	6,694	6,694	-
i320-225	320	102080	33	6,691	6,701	6,701	6,701	-	6,701	6,701	-
i320-231	320	1280	33	9,862	10,242	10,326	10,091	-	10,142	9,863	-
i320-232	320	1280	33	9,933	10,630	10,508	10,116	-	10,183	10,090	-
i320-233	320	1280	33	9,787	10,228	10,289	9,961	-	9,988	9,888	-
i320-234	320	1280	33	9,517	9,851	10,147	9,611	-	9,605	9,520	-
i320-235	320	1280	33	9,945	10,394	10,357	9,945	-	10,118	9,945	-
i320-241	320	20416	33	7,027	7,209	7,162	7,027	-	7,027	7,056	-
i320-242	320	20416	33	7,072	7,159	7,153	7,138	-	7,140	7,135	-
i320-243	320	20416	33	7,044	7,092	7,152	7,113	-	7,132	7,097	-
i320-244	320	20416	33	7,078	7,133	7,191	7,165	-	7,129	7,131	-
i320-245	320	20416	33	7,046	7,133	7,131	7,066	-	7,084	7,081	-
i320-301	320	960	79	23,279	23,621	29,059	23,524	-	24,925	23,367	-
i320-302	320	960	79	23,387	24,152	28,655	23,742	-	24,684	23,652	-
i320-303	320	960	79	22,693	23,140	28,646	22,986	-	24,345	22,787	-
i320-304	320	960	79	23,451	24,182	28,930	23,635	-	24,970	23,554	-
i320-305	320	960	79	22,547	22,923	27,089	22,663	-	23,487	22,550	-
i320-311	320	3690	79	17,945	18,364	20,230	18,909	-	18,642	18,605	-
i320-312	320	3690	79	18,122	18,567	20,488	18,831	-	19,121	18,783	-
i320-313	320	3690	79	17,991	18,161	20,341	18,809	-	18,777	18,440	-
i320-314	320	3690	79	18,088	18,493	20,139	18,770	-	19,141	18,668	-
i320-315	320	3690	79	17,987	18,380	20,293	18,849	-	19,223	18,616	-
i320-321	320	102080	79	15,648	15,725	15,727	15,687	-	15,708	15,687	-
i320-322	320	102080	79	15,646	15,711	15,697	15,695	-	15,693	15,693	-
i320-323	320	102080	79	15,654	15,688	15,693	15,698	-	15,685	15,698	-
i320-324	320	102080	79	15,667	15,756	15,714	15,713	-	15,712	15,709	-
i320-325	320	102080	79	15,649	15,739	15,724	15,711	-	15,712	15,704	-
i320-331	320	1280	79	21,517	22,036	25,752	22,132	-	23,171	22,099	-
i320-332	320	1280	79	21,674	22,109	25,979	22,105	-	23,199	22,027	-
i320-333	320	1280	79	21,339	21,877	25,817	22,083	-	23,075	21,695	-
i320-334	320	1280	79	21,415	21,779	26,403	21,950	-	23,132	21,767	-
i320-335	320	1280	79	21,378	21,839	25,364	21,839	-	22,836	21,622	-
i320-341	320	20416	79	16,296	16,474	17,050	16,592	-	16,638	16,460	-
i320-342	320	20416	79	16,228	16,408	16,865	16,678	-	16,475	16,347	-
i320-343	320	20416	79	16,281	16,482	16,967	16,590	-	16,655	16,582	-
i320-344	320	20416	79	16,295	16,474	16,991	16,643	-	16,660	16,587	-
i320-345	320	20416	79	16,289	16,500	16,897	16,609	-	16,587	16,471	-
i640-001	640	1920	8	4,033	4,033	4,033	4,033	-	4,033	4,033	-
i640-002	640	1920	8	3,588	3,588	3,588	3,588	-	3,588	3,588	-
i640-003	640	1920	8	3,438	3,631	3,438	3,438	-	3,438	3,438	-
i640-004	640	1920	8	4,000	4,169	4,000	4,000	-	4,000	4,000	-
i640-005	640	1920	8	4,006	4,098	4,006	4,006	-	4,006	4,006	-
i640-011	640	8270	8	2,392	2,392	2,392	2,392	-	2,392	2,392	-
i640-012	640	8270	8	2,465	2,466	2,466	2,465	-	2,465	2,465	-
i640-013	640	8270	8	2,399	2,577	2,399	2,399	-	2,399	2,399	-
i640-014	640	8270	8	2,171	2,171	2,171	2,171	-	2,171	2,171	-
i640-015	640	8270	8	2,347	2,351	2,347	2,347	-	2,347	2,347	-
i640-021	640	408960	8	1,749	1,749	1,749	1,749	-	1,749	1,749	-
i640-022	640	408960	8	1,756	1,756	1,756	1,756	-	1,756	1,756	-
i640-023	640	408960	8	1,754	1,754	1,754	1,754	-	1,754	1,754	-

i640-024	640	408960	8	1,751	1,751	1,751	1,751	-	1,751	1,751	-
i640-025	640	408960	8	1,745	1,745	1,745	1,745	-	1,745	1,745	-
i640-031	640	2560	8	3,278	3,591	3,278	3,278	-	3,278	3,278	-
i640-032	640	2560	8	3,187	3,370	3,187	3,187	-	3,187	3,187	-
i640-033	640	2560	8	3,260	3,327	3,260	3,260	-	3,260	3,260	-
i640-034	640	2560	8	2,953	3,091	2,953	2,953	-	2,953	2,953	-
i640-035	640	2560	8	3,292	3,580	3,292	3,292	-	3,292	3,292	-
i640-041	640	81792	8	1,897	1,967	1,897	1,897	-	1,897	1,897	-
i640-042	640	81792	8	1,934	1,976	1,934	1,934	-	1,934	1,934	-
i640-043	640	81792	8	1,931	1,943	1,931	1,931	-	1,931	1,931	-
i640-044	640	81792	8	1,938	1,950	1,938	1,938	-	1,938	1,938	-
i640-045	640	81792	8	1,866	1,866	1,866	1,866	-	1,866	1,866	-
i640-101	640	1920	24	8,764	9,344	9,059	8,773	-	8,778	8,846	-
i640-102	640	1920	24	9,109	9,298	9,478	9,191	-	9,191	9,191	-
i640-103	640	1920	24	8,819	9,188	9,129	8,819	-	8,819	8,848	-
i640-104	640	1920	24	9,040	9,365	9,040	9,084	-	9,040	9,040	-
i640-105	640	1920	24	9,623	9,920	9,823	9,785	-	9,701	9,715	-
i640-111	640	8270	24	6,167	6,232	6,314	6,256	-	6,172	6,189	-
i640-112	640	8270	24	6,304	6,466	6,597	6,400	-	6,424	6,424	-
i640-113	640	8270	24	6,249	6,381	6,486	6,278	-	6,309	6,311	-
i640-114	640	8270	24	6,308	6,323	6,488	6,385	-	6,419	6,308	-
i640-115	640	8270	24	6,217	6,467	6,331	6,228	-	6,233	6,226	-
i640-121	640	408960	24	4,906	4,906	4,906	4,906	-	4,906	4,906	-
i640-122	640	408960	24	4,911	4,911	4,911	4,911	-	4,911	4,911	-
i640-123	640	408960	24	4,913	4,915	4,913	4,913	-	4,913	4,913	-
i640-124	640	408960	24	4,906	4,908	4,906	4,906	-	4,906	4,906	-
i640-125	640	408960	24	4,920	4,920	4,920	4,920	-	4,920	4,920	-
i640-131	640	2560	24	8,097	8,240	8,313	8,201	-	8,216	8,179	-
i640-132	640	2560	24	8,154	8,788	8,500	8,154	-	8,154	8,154	-
i640-133	640	2560	24	8,021	8,279	8,277	8,087	-	8,099	8,084	-
i640-134	640	2560	24	7,754	8,044	7,941	7,754	-	7,825	7,754	-
i640-135	640	2560	24	7,696	8,075	7,927	7,696	-	7,698	7,696	-
i640-141	640	81792	24	5,199	5,309	5,269	5,205	-	5,200	5,200	-
i640-142	640	81792	24	5,193	5,253	5,260	5,210	-	5,232	5,193	-
i640-143	640	81792	24	5,194	5,296	5,252	5,194	-	5,243	5,194	-
i640-144	640	81792	24	5,205	5,307	5,264	5,232	-	5,257	5,234	-
i640-145	640	81792	24	5,218	5,238	5,277	5,277	-	5,273	5,256	-
i640-201	640	1920	49	16,079	16,322	18,124	16,195	-	16,517	16,130	-
i640-202	640	1920	49	16,324	17,236	18,497	16,339	-	16,524	16,409	-
i640-203	640	1920	49	16,124	16,657	18,136	16,389	-	16,721	16,210	-
i640-204	640	1920	49	16,239	16,852	17,735	16,510	-	16,794	16,415	-
i640-205	640	1920	49	16,616	16,787	18,336	16,927	-	17,206	16,643	-
i640-211	640	8270	49	11,984	12,477	12,896	12,222	-	12,542	12,329	-
i640-212	640	8270	49	11,795	12,160	12,746	12,354	-	12,329	12,232	-
i640-213	640	8270	49	11,879	12,128	13,123	12,265	-	12,226	12,077	-
i640-214	640	8270	49	11,898	12,157	12,965	12,352	-	12,391	12,088	-
i640-215	640	8270	49	12,081	12,277	13,142	12,512	-	12,429	12,299	-
i640-221	640	408960	49	9,821	9,864	9,827	9,822	-	9,821	9,821	-
i640-222	640	408960	49	9,798	9,835	9,798	9,807	-	9,798	9,798	-
i640-223	640	408960	49	9,811	9,871	9,858	9,817	-	9,813	9,813	-
i640-224	640	408960	49	9,805	9,851	9,819	9,805	-	9,807	9,817	-
i640-225	640	408960	49	9,807	9,860	9,860	9,815	-	9,815	9,809	-
i640-231	640	2560	49	15,014	15,212	16,687	15,248	-	15,468	15,175	-
i640-232	640	2560	49	14,630	14,945	16,097	15,012	-	15,353	14,877	-
i640-233	640	2560	49	14,797	15,215	16,703	15,087	-	15,297	15,051	-
i640-234	640	2560	49	15,203	15,732	16,942	15,534	-	15,824	15,556	-

i640-235	640	2560	49	14,803	15,514	16,155	15,223	-	15,443	15,024	-
i640-241	640	81792	49	10,230	10,361	10,518	10,431	-	10,355	10,349	-
i640-242	640	81792	49	10,195	10,345	10,437	10,304	-	10,358	10,219	-
i640-243	640	81792	49	10,215	10,348	10,493	10,263	-	10,360	10,355	-
i640-244	640	81792	49	10,246	10,323	10,489	10,369	-	10,397	10,293	-
i640-245	640	81792	49	10,223	10,449	10,410	10,342	-	10,428	10,355	-
i640-301	640	1920	159	45,005	45,656	63,687	46,507	-	53,785	45,506	-
i640-302	640	1920	159	45,736	46,969	65,625	47,042	-	55,674	46,333	-
i640-303	640	1920	159	44,922	45,360	61,364	45,659	-	53,320	45,151	-
i640-304	640	1920	159	46,233	47,152	67,574	47,976	-	56,179	47,221	-
i640-305	640	1920	159	45,902	46,984	64,921	47,555	-	55,740	46,941	-
i640-311	640	8270	159	-	36,527	44,266	38,607	-	41,123	37,760	-
i640-312	640	8270	159	-	36,445	44,709	38,862	-	40,962	37,509	-
i640-321	640	408960	159	31,094	31,228	31,391	31,247	-	31,244	31,205	-
i640-322	640	408960	159	31,068	31,175	31,391	31,233	-	31,307	31,223	-
i640-323	640	408960	159	31,080	31,218	31,420	31,296	-	31,318	31,244	-
i640-324	640	408960	159	31,092	31,219	31,369	31,219	-	31,230	31,227	-
i640-325	640	408960	159	31,081	31,199	31,419	31,273	-	31,247	31,224	-
i640-331	640	2560	159	42,796	43,818	57,927	45,213	-	49,781	44,086	-
i640-332	640	2560	159	42,548	43,830	56,681	44,824	-	51,006	43,918	-
i640-333	640	2560	159	42,345	43,099	58,567	44,781	-	50,785	43,617	-
i640-334	640	2560	159	42,768	43,426	57,959	44,792	-	50,666	43,857	-
i640-335	640	2560	159	43,035	44,118	58,094	44,917	-	51,126	44,168	-
i640-341	640	81792	159	32,042	32,374	34,331	33,033	-	32,868	32,719	-
i640-342	640	81792	159	31,978	32,280	34,185	33,121	-	33,081	32,519	-
i640-343	640	81792	159	32,015	32,328	34,217	33,053	-	33,031	32,738	-
i640-344	640	81792	159	31,991	32,354	34,226	33,172	-	32,952	32,675	-
i640-345	640	81792	159	31,994	32,319	34,206	33,161	-	32,911	32,752	-
lin01	53	160	3	503	503	503	503	503	503	503	503
lin02	55	164	5	557	557	557	557	557	557	557	557
lin03	57	168	7	926	926	926	926	926	926	926	926
lin04	157	532	5	1,239	1,307	1,239	1,239	1,239	1,239	1,239	1,239
lin05	160	538	8	1,703	1,709	1,703	1,703	1,703	1,703	1,703	1,703
lin06	165	548	13	1,348	1,383	1,348	1,348	1,348	1,348	1,348	1,348
lin07	307	1052	5	1,885	1,897	1,885	1,885	1,885	1,885	1,885	1,885
lin08	311	1060	9	2,248	2,252	2,248	2,248	2,248	2,248	2,248	2,248
lin09	313	1064	11	2,752	2,815	2,752	2,752	2,752	2,752	2,752	2,752
lin10	321	1080	19	4,132	4,294	4,149	4,132	4,132	4,135	4,132	4,132
lin11	816	2920	9	4,280	4,323	4,280	4,280	4,280	4,280	4,280	4,280
lin12	818	2924	11	5,250	5,356	5,250	5,250	5,250	5,250	5,250	5,250
lin13	822	2932	15	4,609	4,633	4,609	4,609	4,609	4,609	4,609	4,609
lin14	828	2944	21	5,824	6,143	5,967	5,824	5,824	5,824	5,824	5,824
lin15	840	2968	33	7,145	7,427	7,800	7,145	7,145	7,145	7,145	7,145
lin16	1981	7266	11	6,618	6,696	6,618	6,618	6,618	6,618	6,618	6,618
lin17	1989	7282	19	8,405	8,970	8,405	8,405	8,405	8,405	8,405	8,405
lin18	1994	7292	24	9,714	10,245	10,010	9,724	9,714	9,714	9,720	9,714
lin19	2010	7324	40	13,268	13,632	15,475	13,338	13,338	13,360	13,268	13,268
lin20	3675	13418	10	6,673	7,175	6,673	6,673	6,673	6,673	6,673	6,673
lin21	3683	13434	19	9,143	9,498	9,220	9,143	9,143	9,143	9,143	9,143
lin22	3692	13452	27	10,519	10,740	11,034	10,519	10,519	10,519	10,519	10,519
lin23	3716	13500	51	17,560	18,288	21,763	17,560	17,615	18,480	17,585	17,585

REFERENCES

[1] E. Aarts and J. Korst, "Simulated annealing and boltzmann machines," 1988.

[2] B. G. Agee and M. C. Bromberg, *Method and apparatus for optimization of wireless multipoint electromagnetic communication networks*, US Patent 7,248,841, 2007.

[3] M. P. de Aragão, E. Uchoa, and R. F. Werneck, "Dual heuristics on the exact solution of large steiner problems," *Electronic Notes in Discrete Mathematics*, vol. 7, pp. 150–153, 2001.

[4] E. Balas, "Disjunctive programming: Properties of the convex hull of feasible points," *Discrete Applied Mathematics*, vol. 89, no. 1-3, pp. 3–44, 1998.

[5] M. Bern and D. Bienstock, "Polynomially solvable special cases of the steiner problem in planar networks," *Annals of Operations Research*, vol. 33, no. 6, pp. 403–418, 1991.

[6] M. Bern and P. Plassmann, "The steiner problem with edge lengths 1 and 2," *Information Processing Letters*, vol. 32, no. 4, pp. 171–176, 1989.

[7] D. Bertsimas, E. Litvinov, X. A. Sun, J. Zhao, and T. Zheng, "Adaptive robust optimization for the security constrained unit commitment problem," *IEEE transactions on power systems*, vol. 28, no. 1, pp. 52–63, 2012.

[8] D. Bertsimas, J. Tsitsiklis, *et al.*, "Simulated annealing," *Statistical science*, vol. 8, no. 1, pp. 10–15, 1993.

[9] L. J. Billera, S. P. Holmes, and K. Vogtmann, "Geometry of the space of phylogenetic trees," *Advances in Applied Mathematics*, vol. 27, no. 4, pp. 733–767, 2001.

[10] A. Borchers and D.-Z. Du, "The k-steiner ratio in graphs," *SIAM Journal on Computing*, vol. 26, no. 3, pp. 857–869, 1997.

[11] M. Bordewich and C. Semple, "On the computational complexity of the rooted subtree prune and regraft distance," *Annals of combinatorics*, vol. 8, no. 4, pp. 409–423, 2005.

[12] D. Bryant, "The splits in the neighborhood of a tree," *Annals of Combinatorics*, vol. 8, no. 1, pp. 1–11, 2004.

[13] J. Byrka, F. Grandoni, T. Rothvoss, and L. Sanità, "Steiner tree approximation via iterative randomized rounding," *Journal of the ACM (JACM)*, vol. 60, no. 1, p. 6, 2013.

[14] P. J. Carrington, J. Scott, and S. Wasserman, *Models and methods in social network analysis*. Cambridge university press, 2005, vol. 28.

[15] D. Chakrabarty, J. Könemann, and D. Pritchard, "Hypergraphic lp relaxations for steiner trees," in *International Conference on Integer Programming and Combinatorial Optimization*, Springer, 2010, pp. 383–396.

[16] M. Charikar, C. Chekuri, T.-y. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li, "Approximation algorithms for directed steiner problems," *Journal of Algorithms*, vol. 33, no. 1, pp. 73–91, 1999.

[17] H. Chen and D. D. Yao, "Dynamic scheduling of a multiclass fluid network," *Operations Research*, vol. 41, no. 6, pp. 1104–1115, 1993.

[18] J. Chen*, C.-W. Chen, C.-Y. Tai, and J. Tyan, "Dynamic state-dependent dispatching for wafer fabrication," *International Journal of Production Research*, vol. 42, no. 21, pp. 4547–4562, 2004.

[19] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2009, pp. 199–208.

[20] M. Chlebík and J. Chlebíková, "The steiner tree problem on graphs: Inapproximability results," *Theoretical Computer Science*, vol. 406, no. 3, pp. 207–214, 2008.

[21] M. Conforti, G. Cornuéjols, and G. Zambelli, *Integer Programming*. Springer, 2014.

[22] D. Connors, G. Feigin, and D. Yao, "Scheduling semiconductor lines using a fluid network model," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 2, pp. 88–98, 1994.

[23] Z. J. Czech and P. Czarnas, "Parallel simulated annealing for the vehicle routing problem with time windows," in *Proceedings 10th Euromicro workshop on parallel, distributed and network-based processing*, IEEE, 2002, pp. 376–383.

[24] R. M. Dabbas, H.-N. Chen, J. W. Fowler, and D. Shunk, "A combined dispatching criteria approach to scheduling semiconductor manufacturing systems," *Computers & industrial engineering*, vol. 39, no. 3-4, pp. 307–324, 2001.

[25] R. M. Dabbas and J. W. Fowler, "A new scheduling approach using combined dispatching criteria in wafer fabs," *IEEE Transactions on semiconductor manufacturing*, vol. 16, no. 3, pp. 501–510, 2003.

[26] J. G. Dai *et al.*, "On positive harris recurrence of multiclass queueing networks: A unified approach via fluid limit models," *The Annals of Applied Probability*, vol. 5, no. 1, pp. 49–77, 1995.

[27] G Dantzig and D. R. Fulkerson, "On the max flow min cut theorem of networks," *Linear inequalities and related systems*, vol. 38, pp. 225–231, 2003.

[28] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[29] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical programming*, vol. 91, no. 2, pp. 201–213, 2002.

[30] S. E. Dreyfus and R. A. Wagner, "The steiner problem in graphs," *Networks*, vol. 1, no. 3, pp. 195–207, 1971.

[31] C. Duin, "Preprocessing the steiner problem in graphs," in *Advances in Steiner Trees*, Springer, 2000, pp. 175–233.

[32] C. Duin and S. Voβ, "Efficient path and vertex exchange in steiner tree algorithms," *Networks: An International Journal*, vol. 29, no. 2, pp. 89–105, 1997.

[33] C. W. Duin and A. Volgenant, "Reduction tests for the steiner problem in graphs," *Networks*, vol. 19, no. 5, pp. 549–567, 1989.

[34] C. W. Duin, "Steiner's problem in graphs: Approximation, reduction, variation," PhD thesis, Institute ofActuarial Science and Economic, University of Amsterdam, 1994.

[35] J. Edmonds, "Maximum matching and a polyhedron with 0, 1-vertices," *Journal of research of the National Bureau of Standards B*, vol. 69, no. 125-130, pp. 55–56, 1965.

[36] ——, "Optimum branchings," *Journal of Research of the National Bureau of Standards*, vol. 71, pp. 233–240, 1967.

[37] J. Edmonds and R. Giles, "A min-max relation for submodular functions on graphs," in *Annals of Discrete Mathematics*, vol. 1, Elsevier, 1977, pp. 185–204.

[38] M. J. Feizollahi, M. Costley, S. Ahmed, and S. Grijalva, "Large-scale decentralized unit commitment," *International Journal of Electrical Power & Energy Systems*, vol. 73, pp. 97–106, 2015.

[39] A. E. Feldmann, J. Könemann, N. Olver, and L. Sanità, "On the equivalence of the bidirected and hypergraphic relaxations for steiner tree," *Mathematical programming*, vol. 160, no. 1-2, pp. 379–406, 2016.

[40] A. E. Feldmann and D. Marx, "The complexity landscape of fixed-parameter directed steiner network problems," *arXiv preprint arXiv:1707.06808*, 2017.

[41] J. Felenstein, *Inferring phylogenies*. Sinauer associates Sunderland, MA, 2004, vol. 2.

[42] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," in *Classic papers in combinatorics*, Springer, 2009, pp. 243–248.

[43] Z.-H. Fu and J.-K. Hao, "Swap-vertex based neighborhood for steiner tree problems," *Mathematical Programming Computation*, vol. 9, no. 2, pp. 297–320, 2017.

[44] D. Gamarnik, "Fluid models of queueing networks," *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[45] R. Gamberini, F. Lolli, B. Rimini, M. Torelli, and E. Castagnetti, "An innovative approach for job pre-allocation to parallel unrelated machines in the case of a batch sequence dependent manufacturing environment," *International Journal of Production Research*, vol. 49, no. 14, pp. 4353–4376, 2011.

[46] M. X. Goemans, "The steiner tree polytope and related polyhedra," *Mathematical programming*, vol. 63, no. 1-3, pp. 157–182, 1994.

[47] M. X. Goemans and Y.-S. Myung, "A catalog of steiner tree formulations," *Networks*, vol. 23, no. 1, pp. 19–28, 1993.

[48] M. X. Goemans, N. Olver, T. Rothvoß, and R. Zenklusen, "Matroids and integrality gaps for hypergraphic steiner tree relaxations," in *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, ACM, 2012, pp. 1161–1176.

[49] D. R. Greening, "Parallel simulated annealing techniques," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1-3, pp. 293–306, 1990.

[50] E. Halperin and R. Krauthgamer, "Polylogarithmic inapproximability," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, ACM, 2003, pp. 585–594.

[51] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[52] S. Hougardy, J. Silvanus, and J. Vygen, "Dijkstra meets steiner: A fast exact goal-oriented steiner tree algorithm," *Mathematical Programming Computation*, vol. 9, no. 2, pp. 135–202, 2017.

[53] J. R. Jackson, "Jobshop-like queueing systems," *Management science*, vol. 10, no. 1, pp. 131–142, 1963.

[54] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[55] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*, Springer, 1972, pp. 85–103.

[56] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2003, pp. 137–146.

[57] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[58] T. Koch and A. Martin, "Solving steiner tree problems in graphs to optimality," *Networks: An International Journal*, vol. 32, no. 3, pp. 207–232, 1998.

[59] T. Koch, A. Martin, and S. Voß, "Steinlib: An updated library on steiner tree problems in graphs," in *Steiner trees in industry*, Springer, 2001, pp. 285–325.

[60] J. Könemann, D. Pritchard, and K. Tan, "A partition-based relaxation for steiner trees," *Mathematical Programming*, vol. 127, no. 2, pp. 345–370, 2011.

[61] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*, vol. 7, no. 1, pp. 48–50, 1956.

[62] P. Kuehn, "Approximate analysis of general queuing networks by decomposition," *IEEE Transactions on communications*, vol. 27, no. 1, pp. 113–126, 1979.

[63] M. E. Kuhl and G. R. Laubisch, "A simulation study of dispatching rules and rework strategies in semiconductor manufacturing," in *2004 IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop (IEEE Cat. No. 04CH37530)*, IEEE, 2004, pp. 325–329.

[64] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[65] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European journal of operational research*, vol. 59, no. 3, pp. 345–358, 1992.

[66] J. K. Lenstra and A. R. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 221–227, 1981.

[67] S. Li, T. Tang, and D. W. Collins, "Minimum inventory variability schedule with applications in semiconductor fabrication," *IEEE Transactions on Semiconductor Manufacturing*, vol. 9, no. 1, pp. 145–149, 1996.

[68] Y. Li, Z. Jiang, and W. Jia, "An integrated release and dispatch policy for semiconductor wafer fabrication," *International Journal of Production Research*, vol. 52, no. 8, pp. 2275–2292, 2014.

[69] ——, "Api-based two-dimensional dispatching decision-making approach for semiconductor wafer fabrication with operation due date–related objectives," *International Journal of Production Research*, vol. 55, no. 1, pp. 79–95, 2017.

[70] Á. Lorca, X. A. Sun, E. Litvinov, and T. Zheng, "Multistage adaptive robust optimization for the unit commitment problem," *Operations Research*, vol. 64, no. 1, pp. 32–51, 2016.

[71] A. Lorca and X. A. Sun, "Adaptive robust optimization with dynamic uncertainty sets for multi-period economic dispatch under significant wind," *IEEE Transactions on Power Systems*, vol. 30, no. 4, pp. 1702–1713, 2014.

[72] S. C. Lu, D. Ramaswamy, and P. Kumar, "Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants," *IEEE Transactions on Semiconductor Manufacturing*, vol. 7, no. 3, pp. 374–388, 1994.

[73] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297.

[74] T. L. Magnanti and R. T. Wong, "Network design and transportation planning: Models and algorithms," *Transportation science*, vol. 18, no. 1, pp. 1–55, 1984.

[75] R. K. Martin, R. L. Rardin, and B. A. Campbell, "Polyhedral characterization of discrete dynamic programming," *Operations Research*, vol. 38, no. 1, pp. 127–138, 1990.

[76] H.-S. Min and Y. Yih, "Selection of dispatching rules on multiple dispatching decision points in real-time scheduling of a semiconductor wafer fabrication system," *International Journal of Production Research*, vol. 41, no. 16, pp. 3921–3941, 2003.

[77] E. MING-IHsieh and M Tsai, "Fasterdsp: A faster approximation algorithm for directed steiner tree problem," *Journal of information science and Engineering*, vol. 22, pp. 1409–1425, 2006.

[78] M. Mittler and A. K. Schoemig, "Comparison of dispatching rules for semiconductor manufacturing using large facility models," in *WSC'99. 1999 Winter Simulation Conference Proceedings.'Simulation-A Bridge to the Future'(Cat. No. 99CH37038)*, IEEE, vol. 1, 1999, pp. 709–713.

[79] L. Mönch, J. W. Fowler, S. Dauzère-Pérès, S. J. Mason, and O. Rose, "A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations," *Journal of scheduling*, vol. 14, no. 6, pp. 583–599, 2011.

[80] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[81] T. Polzin and S. V. Daneshmand, "A comparison of steiner tree relaxations," *Discrete Applied Mathematics*, vol. 112, no. 1-3, pp. 241–261, 2001.

[82] ——, "Extending reduction techniques for the steiner tree problem," in *European Symposium on Algorithms*, Springer, 2002, pp. 795–807.

[83] ——, "Improved algorithms for the steiner problem in networks," *Discrete Applied Mathematics*, vol. 112, no. 1-3, pp. 263–300, 2001.

[84] ——, "On steiner trees and minimum spanning trees in hypergraphs," *Operations Research Letters*, vol. 31, no. 1, pp. 12–20, 2003.

[85] R. C. Prim, "Shortest connection networks and some generalizations," *The Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

[86] S. Rajagopalan and V. V. Vazirani, "On the bidirected cut relaxation for the metric steiner tree problem.," in *SODA*, vol. 99, 1999, pp. 742–751.

[87] D. J. Ram, T. Sreenivas, and K. G. Subramaniam, "Parallel simulated annealing algorithms," *Journal of parallel and distributed computing*, vol. 37, no. 2, pp. 207–212, 1996.

[88] D. Rehfeldt, "A generic approach to solving the steiner tree problem and variants," 2015.

[89] O. Rose, "Scheduling and dispatching: Some issues of the critical ratio dispatch rule in semiconductor manufacturing," in *Proceedings of the 34th conference on Winter simulation: exploring new frontiers*, Winter Simulation Conference, 2002, pp. 1401–1405.

[90] ——, "The shortest processing time first (sptf) dispatch rule and some variants in semiconductor manufacturing," in *Proceeding of the 2001 Winter Simulation Conference (Cat. No. 01CH37304)*, IEEE, vol. 2, 2001, pp. 1220–1224.

[91] T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro, "A stochastic programming approach for supply chain network design under uncertainty," *European Journal of Operational Research*, vol. 167, no. 1, pp. 96–115, 2005.

[92] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media, 2003, vol. 24.

[93] J. G. Shanthikumar and J. A. Buzacott, "Open queueing network models of dynamic job shops," *The International Journal Of Production Research*, vol. 19, no. 3, pp. 255–266, 1981.

[94] J. G. Shanthikumar, S. Ding, and M. T. Zhang, "Queueing theory for semiconductor manufacturing systems: A survey and open problems," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 4, pp. 513–522, 2007.

[95] M. Siebert, S. Ahmed, and G. Nemhauser, "A linear programming based approach to the steiner tree problem with a fixed number of terminals," *Networks*, 2019. DOI: 10.1002/net.21913. [Online]. Available: https://doi.org/10.1002/net.21913.

[96] M. Siebert, K. Bartlett, H. Kim, S. Ahmed, J. Lee, D. Nazzal, G. Nemhauser, and J. Sokol, "Lot targeting and lot dispatching decision policies for semiconductor manufacturing: Optimisation under uncertainty with simulation validation," *International Journal of Production Research*, vol. 56, no. 1-2, pp. 629–641, 2018.

[97] Y. S. Song, "On the combinatorics of rooted binary phylogenetic trees," *Annals of Combinatorics*, vol. 7, no. 3, pp. 365–379, 2003.

[98] R. Srikant and L. Ying, *Communication networks: an optimization, control, and stochastic networks perspective*. Cambridge University Press, 2013.

[99] R. P. Stanley, *Enumerative Combinatorics*. Cambridge University Press, 1999, vol. 2.

[100] H. Takahashi, "An approximate solution for the steiner problem in graphs," *Math. Japonica.*, vol. 6, pp. 573–577, 1990.

[101] P. Toth and D. Vigo, *The vehicle routing problem*. SIAM, 2002.

[102] J. C. Tyan, J. C. Chen, and F.-K. Wang, "Development of a state-dependent dispatch rule using theory of constraints in near-real-world wafer fabrication," *Production Planning & Control*, vol. 13, no. 3, pp. 253–261, 2002.

[103] E. Uchoa, M. Poggi de Aragão, and C. C. Ribeiro, "Preprocessing steiner problems from vlsi layout," *Networks: An International Journal*, vol. 40, no. 1, pp. 38–50, 2002.

[104] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*, Springer, 1987, pp. 7–15.

[105] F. Wang, X. Lai, and N. Shi, "A multi-objective optimization for green supply chain network design," *Decision Support Systems*, vol. 51, no. 2, pp. 262–269, 2011.

[106] D. M. Warme and J. S. Salowe, *Spanning trees in hypergraphs with applications to Steiner trees*. University of Virginia, 1998.

[107] D. Watel and M.-A. Weisser, "A practical greedy approximation for the directed steiner tree problem," *Journal of Combinatorial Optimization*, vol. 32, no. 4, pp. 1327–1370, 2016.

[108] P. Winter, "Reductions for the rectilinear steiner tree problem," *Networks*, vol. 26, no. 4, pp. 187–198, 1995.

[109] R. T. Wong, "A dual ascent approach for steiner tree problems on a directed graph," *Mathematical programming*, vol. 28, no. 3, pp. 271–287, 1984.

[110] M.-C. Wu, Y. Huang, Y. Chang, and K. Yang, "Dispatching in semiconductor fabs with machine-dedication features," *The International Journal of Advanced Manufacturing Technology*, vol. 28, no. 9-10, pp. 978–984, 2006.

[111] C. Xie and T. T. Allen, "Simulation and experimental design methods for job shop scheduling with material handling: A survey," *The International Journal of Advanced Manufacturing Technology*, vol. 80, no. 1-4, pp. 233–243, 2015.

[112] R. Xu and D. C. Wunsch, "Survey of clustering algorithms," 2005.